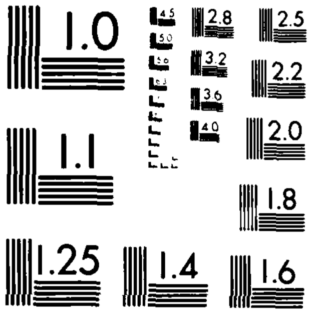
AD-A146 249 ON LINE DIGITIZER SOFTWARE(U) ARMY MISSILE COMMAND 1/2
REDSTONE ARSENAL AL ADVANCED SENSORS DIRECTORATE
S R SIMS JUN 84 DRSMI/RE-84-17-TR SBI-AD-E950 550
UNCLASSIFIED F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

(12)

AD-A146 249

TECHNICAL REPORT RE-84-17

ON LINE DIGITIZER SOFTWARE

S. Richard F. Sims
Advanced Sensors Directorate
US Army Missile Laboratory

JUNE 1984

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

84 09 13 073

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RE-84-17	2. GOVT ACCESSION NO. [illegible]	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) On Line Digitizer Software		5. TYPE OF REPORT & PERIOD COVERED Interim Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) S. Richard F. Sims		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Commander, US Army Missile Command ATTN: DRSMI-RES Redstone Arsenal, AL 35898		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander, US Army Missile Command ATTN: DRSMI-RPT Redstone Arsenal, AL 35898		12. REPORT DATE June 1984
		13. NUMBER OF PAGES 165
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Video digitizing and recording
Digital video recording and playback

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the software developed to date for the On Line digitizer system. The software provides means to transfer digital imagery and auxilary data to a host computer hard disk, to test system components and display the imagery and auxilary data in various formats. The system digitizes analog video (RS-170 or RS-343) at a 10 MHz rate to eight bits and records 16 auxilary analog channels for approximately 15 minutes on 28 track tape.

TABLE OF CONTENTS

APPENDIXES

APPENDIXES (CONTINUED)

✓

A1

1.0 INTRODUCTION

The ON LINE DIGITIZER is a digital data recording/playback system capable of recording real time video for approximately 15 minutes continuously along with 16 auxiliary channels and one digital channel. The main input is either standard RS-170 or RS-343 video which is digitized at 10.08 MHZ and phase locked to video horizontal sync. The first 512 samples of the 521 digitized on each line are stored in the ON LINE DIGITIZER frame buffer (AVA). The 16 auxiliary channels are -10. to +10. volt analog input that are digitized to 12 bits and sampled at the (video horizontal sync rate)/16. The digital channel is RS-232 input. All data other than the video is stored during the field into a memory and transfered to AVA frame buffer memory during the video vertical interval.

The ON LINE DIGITIZER consists of two major subsystems, the airborne unit and the ground unit. The airborne unit contains an Ampex AR-1700 28 track digital tape recorder with a custom digital processing unit. The airborne unit is for use in aircraft, range, or laboratory data recording. The ground unit contains an Ampex HBR-3000 28 track digital tape recorder with Datum IRIG search unit and digital image frame buffer. The ground unit takes the recorded 28 track tapes and plays back the data at selected rates for review and transfer of data to the computer. The image frame buffer contains memory area to hold 4 sequential video fields (512 pixels by 240 lines), 256 auxiliary words and frame IRIG time. This report describes and lists software utilities for use with the ON LINE DIGITIZER ground unit which are necessary to transfer data in various modes of operation and diagnostic utilities for hardware testing.

The two subsystems of the ON LINE DIGITIZER can be connected together in a real time mode bypassing the tape drives. This mode is used in diagnostic tests and for single snap shot digitization of images.

The ON LINE DIGITIZER has been in a continuous state of hardware upgrade and software development since its installation in the Sensor Signal Processing System (SSPS) and will continue to be modified to add additional improvements and capability which may impact execution of the software described in this report.

2.Ø GENERAL USAGE AND PROCESS QUOTAS

This following software can ONLY be run under the AVA username. The process quotas under AVA have been set to allow large buffered I/O transfers.

The AVA frame buffer has four fields stored at a time. The dip switch on card 13 in the frame buffer housing allows display of fields Ø and 1 individually in realtime. Field 2 and 3 can also be displayed but only combined with Ø and 1 respectively. This dip switch along with the master toggle switch should be specifically set to allow viewing of the desired data during playback or in real time direct connect mode.

3.Ø HARDWARE/SOFTWARE INTERFACES

The On line digitizer frame buffer (AVA) uses a programed I/O interface to the UNIBUS on a VAX 11/78Ø. This unique device does not have DMA capability. The interface also requires total bus control during the I/O transfer. It will not tolerate things like interval clock interrupts, etc. and therefore the driver raises the interrupt priority level to "handle" this. The only after effect is the CPU clock and other things waiting for the I/O driver to release the CPU and UNIBUS do not get serviced and of course all other processes have to wait until the I/O is complete. The software driver AVDRIVER is listed in the appendix.

The On line digitizer search unit is interfaced to the VAX UNIBUS with a standard DEC DR11-C. This unit can control all ground unit functions or can be remotely under software control. The software driver ODDRIVER is listed in the appendix.

4.Ø SOFTWARE UTILITIES FOR IMAGE TRANSFER

4.1 Images To Disk

The following programs are to be used when large amounts of disk space are available. Large being defined as enough contiguous space to hold the number of desired images which can be calculated as follows:

```
BLOCKS=NI+((NI*NC*NR*NB)/512)
```

```
where   BLOCKS=NUMBER OF CONTIGUOUS BLOCKS OF DISK SPACE NEEDED (512 BYTES PER BLOCK)
        NI=NUMBER OF IMAGES
        NC=NUMBER OF COLUMNS IN THE IMAGE (Horizontal Picture elements)
        NR=NUMBER OF ROWS    IN THE IMAGE (Vertical   Picture elements)
        NB=NUMBER OF BYTES PER PICTURE ELEMENT (normally 8 bits/pixel)
```

4.1.1 Single Image To Disk : [DISK]AVATODSK2 -

The image to disk program will transfer the current image in the AVA frame buffers Ø and 1 to a complete frame in contiguous image disk format. Note that this is true if the frame is "frozen" or not. The normal mode is to freeze the frame using the STOP control on the ground unit and then execute the program to transfer the image. The disk format is transfered easily to tape witn @DISK$USERDISK:[SUBIMAGE]SUBNATO. The user will be asked to enter the disk storage name. The standard convention used for disk image names is described as follows:

XyyyyZZZZ.IMG

where

X= Alpha character
y= Sequence number
Z= Subimage sequence number (i.e. Subimage of Xyyyy)

4.2 Sequential Images To Disk : [MAXDISK]MTODSK3

Sequential images can be transfered from the HBR-3ØØØ during 32 to 1 playback or 3 3/4 speed to disk. MTODSK3 checks the disk and allows transfer only after it knows how many contiguous blocks are available for image storage. The images are placed in IMAGES.DAT. Only fields Ø and 1 are transfered to disk. Fields 2 and 3 are skipped to allow time for field Ø and 1 transfer completion. This of course means only every other frame is transfered to disk.

4.3 Sequential Images To Disk : [MAXDISK]MTODSK3A

This program performs the same funtions as MTODSK3 however in addition the frame buffer IRIG time is also stored in memory for each image. After all images have been transfered to disk the stored IRIG times are written to DISK$AVA:[AVA]IRIGS.DAT.

4.4 Sequential Images To Disk Plus : [MAXDISK]MTODSK4

Sequential images can be transfered from the HBR-3000 during 32 to 1 playback or 3 3/4 speed to disk. MTODSK4 checks the disk and allows transfer only after it knows how many contiguous blocks are available for image storage. The images are placed in IMAGES.DAT. The program puts out the maximum number of fields to disk possible. This is more data than MTODSK3 will transfer since it checks to see if I/O is complete and then tranfers the next available field regardless of which one it is.

4.4.1 Initial And Subsequent Runs Of MTODSK3,MTODSK3A Or MTODSK4 -

The initial run of MTODSK3,MTODSK3A or MTODSK4 will not start I/O transmission immediately after the beginning of the run. The disk space is interrogated to decide what is the maximum contiguous space available. In order for the entire disk space to be usable contiguously, the disk pack must be initialized with the /INDEX=BEGINNING qualifier. Interrogation may take several seconds before the search is complete. After the largest space is found the file is opened and the area is allocated. The size of this allocated area and hence the number of images which can be written is highly dependant on the specific medium (disk pack) used in the disk drive. Individual disk packs have different characteristics one of which is where the bad blocks, if any, are located. If very long sequences of data are needed to be transfered several packs may have to be checked before actual execution. After the IMAGES.DAT file has been created by the initial run of MTODSK3,MTODSK3A or MTODSK4 the file will be over written by any subsequent running of these programs. Therefore if the data on the disk in IMAGES.DAT is needed in this form another disk pack is required before the subsequent runs.

5.0 UTILITIES FOR DISPLAY OF IMAGES ON DISK

5.1 Display Of A Single Image From Disk : @SUBIMAGE

The subimage data base software uses the SSPS standard image file format which consists of a contiguous image file and associated header file with the same name. The subimage data base software is executed by the command @SUBIMAGE. The operator enters the following answers to program questions:

```
        1. Is a new image list file required?          NO
        2. Is image from disk or tape? (D or T)        D
```

The operator then enters the following commands:

```
        1.    L    -     tells the program you want to load an image file
        2.         -     "enter the image file name here XYYYYZZZZ.IMG"
        3.    Z    -     tells the program you want to display loaded image
        4.  <CR>   -     just enter carriage return for the AGC value

        ........The image is now displayed on the Grinnell ..........

        5.    N    -     tells the program you want to go to the next image
        6.               GO TO 1.
```

5.2 Display Of IMAGES.DAT

IMAGES.DAT is generated by MTODSK3,MTODSK3A, or MTODSK4 on DISK$AVA:[AVA]. The file is one large contiguous set of images 512 by 240 pixels per field. To transfer IMAGES.DAT to the GRINNELL one of the following routines can be used:

* [MAXDISK]MDSKTOGRN - Transfers full frames of images to the GRINNELL field at a time.

* [MAXDISK]MDTOGRN - Transfers full frames of images to the GRINNELL frame at a time.

* [MAXDISK]FIELDSGRN - Transfers fields as one image to the GRINNELL field at a time.

6.0 IMAGES.DAT TO SUBIMAGE DISK FORMAT [MAXDISK]MDSKTOFIL

MDSKTOFIL takes the IMAGES.DAT file and transfers, from the desired starting image, each frame to a unique image file in SSPS standard image format with a header file. The number of images to transfer is an input as well as the increment between frames. The IMAGES.DAT can be created by *MTODSK3,MTODSK3A, or MTODSK4.*

7.Ø IMAGES.DAT TO SUBIMAGE DISK FORMAT [MAXDISK]MDSKTFIL2

This program performs the same function as *MDSKTOFIL but in* addition reads the DISK$AVA:[AVA]IRIGS.DAT file and places the IRIG time in the NATO header two for each image header file. The IRIGS.DAT file is created by MTODSK3A automatically, however, if the IRIG times are not available the IRIGS.DAT file can also be generated by "other" means.

8.Ø AVA FRAME BUFFER UTILITIES

8.1 AVA FIELDS TO GRINNELL. [AVA]AVAFIELDS

AVAFIELDS displays the current field desired residing in the AVA frame buffer on the GRINNELL. The operator input is the field number Ø,1,2, or 3. The operator is most cases will "freeze" the video before transfering the field image by using the STOP control on the ground unit. The fields are loaded by the hardware in sequence first field Ø then 1, 2 and 3 and the the process is repeated as long as input data continues.

[AVA]AVAGROUP8 reads the current AVA fields Ø and 1 and puts them on the GRINNELL updated field at a time continuously.

[AVA]AVAGROUP9 is the same program as AVAGROUP8 except the block sizes are as large as possible for AVA frame buffer reads.

8.2 Writing To AVA Field Memory. [AVA]AVAFWRITE

AVAFWRITE allows the user to write to a specific field memory area in the AVA frame buffer. The data written is an input to the program therefore 256 values can be designated. The same value is written over the entire specified field memory area.

9.Ø INPUT/OUTPUT TIMING

9.1 Writing A Ramp Pattern To AVA Memory [AVA]RAMPMAX

RAMPMAX writes a double ramp grayscale to the AVA memory in all four fields. This routine demonstrates the write timing to the AVA memory from the VAX 11/78Ø. The frame buffer needs to be stopped to allow only the computer to write into AVA field memory.

9.2 Reading AVA Fields [AVA]RAMPMAX2

RAMPMAX2 reads frame 1 of the AVA MEMORY or fields Ø and 1. This routine demonstrates the read timing from the ava memory to the VAX 11/78Ø.

9.3 HBR - 3ØØØ Data Transfer Timing. [MAXDISK]MTODSKT

MTODSKT will time AVA reads with a variable write delay, for simulation of the disk write time, for N fields. The HBR-3ØØØ speed for reasonable data rates must be at 3 3/4 ips or 1 7/8 ips.

1Ø.Ø AVA FRAME BUFFER MEMORY DIAGNOSTICS

1. FAVAMEMT - 4 pattern test on video memory
2. FAVAMENT2 - 4 pattern test on video memory *
3. AVAMEMT - user entered pattern test
4. AVAMEMT2 - user entered pattern test *
5. AAVAMEMT - 4 pattern test on ALL AVA memory

* - Specific error printouts

11.Ø UTILITIES FOR AUXILIARY DATA TRANSFER

11.1 Reading The 16 Auxiliary Channels (First Set Only)

[AVA]AUX will display the channel number and the voltage input on each of the 16 auxiliary input channels. The display will be up dated by direct cursor addressing of the screen using the channel address in the auxiliary word and no scrolling will occur. The input channels are sampled at the (video horizontal sync rate)/16 and can range from -1Ø. to +1Ø. volts. This program reads only the first 16 words from the auxiliary memory area and therefore will not reflect the actual signal frequency response recorded or being sampled in real time.

[AVA]AUX2 will display the 16 auxiliary channels as does AUX except the screen will scroll and reflect the staggering positions of the data as actually stored in the auxiliary memory. The channel addresses are not used for display of the data and the format is in hexidecimal only. This routine like AUX reads only the first 16 words from the auxiliary memory area.

11.2 Plotting The 16 Auxiliary Channels

[AVA]AUXPLOT will plot all sixteen samples of the requested auxiliary channel. The scale is +1Ø. to -1Ø. volts vertically with the samples plotted horizontally in sequence repeatedly. The data in the AVA frame buffer is read before each plot is generated. The plot instructions use VT-52 escape sequences and therefore require a compatible terminal. Prior to execution of this routine it is necessary to ensure that the terminal is in VT52 mode by executing the following DCL command. SET TERM/VT52. This process should also be followed in AUXPLOTA.

[AVA]AUXPLOTA will plot all sixteen samples of all channels five times across the plot. This is useful in testing auxiliary channel frequency response by connecting all channels to the same varying signal and adjusting the amplitude and rate of the variation.

11.3 Utilities For Reading The IRIG Time

[AVA]AUXIRIG will display the IRIG time placed in the AVA frame buffer. This IRIG time is updated either in real time or by HBR-3000 playback. The display places the IRIG on the screen without scrolling.

[AVA]AUXIRIG2 will scroll the IRIG time on the screen and is used for checks of video sync stability and IRIG read timing.

12.0 UTILITIES FOR HBR-3000 TAPE CONTROL AND SEARCH

12.1 Reading IRIG Time From The Tape Search Unit

[AVA.TAPEDRIVE]IRIGREAD will freeze the IRIG output register at each read and transfer the IRIG to the display and repeat continuously.

12.2 HBR-3000 Drive Control

[AVA.TAPEDRIVE]COMMAND lets the knowledgeable user remotely control the HBR-3000. The search unit REMOTE/LOCAL switch must be in REMOTE otherwise this program has no effect. It is suggested that the user be familiar with the functions in the Datum Search Unit Manual before running this program. Some typical commands are shown below:

NOTE: all commands are in octal for decoding the bit functions

157000	STOP
150001	TRANSLATE IRIG A WITH ZERO FRAME BYPASS
156400	UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
157476	SET THE FILTERS TO 120 IPS
157201	DRIVE FORWARD AT 120 IPS
157221	DRIVE FORWARD AT 240 IPS (FAST FOWARD)
157061	DRIVE FORWARD AT 3 3/4 IPS (32 TO 1)
157222	DRIVE REVERSE AT 240 IPS (FAST REVERSE)
157202	DRIVE REVERSE AT 120 IPS
157206	SINGLE CYCLE SEARCH MODE AT 120 IPS

12.3 IRIG List Generation For Image Tranfers

[AVA.TAPEDRIVE]REVIEW is for generating an IRIG list file of the single images the user wants on disk in disk image format. The user would start REVIEW and then play the tape on the HBR-3ØØØ at 12Ø ips (normal speed). When a desired image appears the user presses a return at the terminal. When a return is pressed the current IRIG time is read from the AVA frame buffer and is written to disk. This is repeated as many times as needed. After reviewing the portion of tape required by the user typically several IRIG times would be in the REVIEW.IRG file, and the user would then type in Z to terminate program execution.

12.4 Transfering IRIG List Images To Disk

[AVA.TAPEDRIVE]RTODISK will use the IRIG list file REVIEW.IRG to scan the tape on the HBR-3ØØØ and transfer the images to disk in disk image format. The user must enter the beginning file name for the first image and there after the file name sequence number will be automatically incremented. Note that there must be enough contiguous disk space available in the default file directory else the program will abort.

12.5 Utilities For The Tape Search Unit

1. [AVA.TAPEDRIVE]STOSTART allows the user to enter an IRIG time to search for. The tape will be transfered to the position where the IRIG time occurs.

2. [AVA.TAPEDRIVE]STATUSR displays the current status of the HBR-3ØØØ repeatedly.

3. [AVA.TAPEDRIVE]IOTEST allows the user to perform a fully functional test on the DR11-C interface tied to the Tape Search Unit. The maintenance cable or equivalent must be connected from the output port to the input port. This program tests all data bits on the DR11-C and the "A" interrupt hardware.

APPENDIX A

[AVA.DISK]AVATODSK2

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM WRITES THE CURRENT AVA FRAME BUFFER IMAGE
C       FIELDS Ø AND 1 ON TO DISK IN SUBIMAGE DATA BASE FORMAT.
C
C       THE IMAGE NAME XXX IS REQUESTED. THIS NAME IS USED FOR THE
C       XXX.IMG FILE AND THE XXX.HDR FILE.
C
C       THIS FILE CAN THEN BE ACCESSED BY ANY OF THE SUBIMAGE DATA BASE
C       SOFTWARE SET.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM IS DIFFERENT FROM AVA1TODSK.FOR IN THAT
C       AVA1TODSK.FOR WRITES TO DISK IN A COMPLETELY DIFFERENT FORMAT AND ONLY
C       TO DISK$IMAGES:[AVA]IMAGES.DAT WHEREAS THIS PROGRAM WRITES IT OUT IN
C       THE SUBIMAGE DATA BASE FORMAT AND WILL USE THE NAME INPUT BY THE USER.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        PARAMETER IEVF = 4
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
        INTEGER*4 IMGADR,SYS$ASSIGN,IMGADR2,SYS$GETMSG
        INTEGER*4 LIB$FREEVM,LIB$GETVM,SYS$DASSGN
        INTEGER AVACHAN
        DIMENSION AR(65,65),BR(65,65)
        CHARACTER*6Ø TITLE,MSGBUF
C       TITLE='WRITE TO DISK TIME FOR ONE 512X48Ø IMAGE'
        DSPSCF=1.Ø
        IMGMAPC(1)=1
        IMGMAPC(2)=1
        I=SYS$ASSIGN('TT',IVTC,,)
        IF(.NOT.I)TYPE *,'ERROR IN TT CHANNELL ASSIGN'
        I=SYS$ASSIGN('GRAØ',GRCHAN,,)
        IF(.NOT.I)THEN
        TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        STOP
        ENDIF
C       CODE TO READ AVA IMAGE INTO VIRTUAL MEMORY
        ISTATUS=SYS$ASSIGN('AVA0',AVACHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA GRCHANNEL ASSIGN'
C       FNAM = 'BARDOTS.IMG'
C       READ(5,5678)FNAM
5678    FORMAT(A80)
C       TYPE *,FNAM
C       CALL READSUB(ILEN,IWD,IMGADR)   ! READ IMAGE FROM DISK
C       IMGMAPC(3)=ILEN
C       IMGMAPC(4)=IWD
C       IMGMAPC(3)=512
C       IMGMAPC(4)=512
C       CALL DSPIMG(%VAL(IMGADR))       ! PUT IMAGE ON THE GRINNELL
        NCOL=512
        NROW=480
        ILEN=NROW
        IWD=NCOL
        IMGMAPC(3)=ILEN !LENGTH OF IMAGE
        IMGMAPC(4)=IWD  !WIDTH  OF IMAGE
        I=IWD
        NBYT=(I+1)*ILEN*2
        I=LIB$GETVM(NBYT,IMGADR)
        IF(.NOT.I)TYPE *,' ERROR IN VIRTUAL MEMORY ASSIGNMENT  1'
        CALL AVAREAD(%VAL(IMGADR),AVACHAN)
        HEAD(8)='       1'      !ONE CHARACTER PER CHANNEL

        I = LIB$GETVM(10000,HDR2ADR)
        IF(.NOT. I) CALL ERRSTOP(I,'ERROR GETTING HDR2 VM','AVATODSK')
        CALL ADDHDR2(%VAL(HDR2ADR))
        HDR2LEN=576
        CURRENTNUMFL=0
        CALL DSPIMG(%VAL(IMGADR))       ! PUT IMAGE ON THE GRINNELL
C       TYPE *,'IWD=',IWD,'  ILEN=',ILEN
        FNAM='SIMS00001.IMG'
        TYPE *,'ENTER OUTPUT FILE NAME. (123456789.IMG)'
        FNAM=' '
        READ(5,123)FNAM
123     FORMAT(A)
C       CALL TIMRB
C       IWD=512
C       ILEN=512
        CALL TODISK(%VAL(IMGADR),IWD,ILEN,AVACHAN)
C       CALL TIMRE
C       CALL HEADER(TITLE)
C       I=IWD
C       NBYT=(I+1)*ILEN*2
C       I=LIB$GETVM(NBYT,IMGADR2)
C       IF(.NOT.I)TYPE *,' ERROR IN VIRTUAL MEMORY ASSIGN FOR OUTPUT IMAGE'
C       I=LIB$GETVM(NBYT,IMGADR3)
C       IF(.NOT.I)TYPE *,' ERROR IN VIRTUAL MENORY ASSIGN FOR OUTPUT IMAGE'
C       FNAM(23:31)='A00030001'
C       CALL READSUB(ILEN,IWD,IMGADR3)  ! READ IMAGE FROM DISK
C       IMGMAPC(3)=ILEN
```

```
C       IMGMAPC(4)=IWD
C       TYPE *,'IWD=',IWD,'  ILEN=',ILEN
        STOP '512X480 IMAGE WRITTEN TO DISK
        1          '
        END
        SUBROUTINE TODISK(IMAGE,IWD,ILEN,AVACHAN)
        EXTERNAL IO$READVBLK
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]AUTOIMG.CMN/NOLIST'
        INTEGER*2 US,TS,UM,TM
        INTEGER*2 UH,TH,UD,TD,HD
        INTEGER*2 MS,HMS,TMS,IOSB(4)
        INTEGER*2 IMAGE(NCOL+1,NROW),HDR2LEN,D(8),X,Y
        INTEGER*4 AUTOWRTSB
        INTEGER AVACHAN,SYS$QIOW
        CHARACTER*10000 HDR2ADR
        CHARACTER*3 MONTH,DAY,YEAR*2,WD*8,LEN*8,TIMEA*8
        CHARACTER*5 IFIRST5,ILAST4*4,TNAME*9
        HDR2LEN=HDR2LEN
C       TYPE *,'HDR2LEN',HDR2LEN
        CALL CNVRT(%VAL(HDR2ADR),HDR2LEN,HDR2ADR)
C       TYPE *,HEAD
        CALL IDATE(IMONTH,IDAY,IYEAR)
        ENCODE(3,200,MONTH)IMONTH
        ENCODE(3,200,DAY  )IDAY
200     FORMAT(I3)
        ENCODE(2,100,YEAR )IYEAR
        HEAD(3)='OLDFAAD '
        HEAD(1)='USAMICOM'
        HEAD(2)=YEAR//MONTH//DAY
100     FORMAT(I2)
        ENCODE(8,200,WD)IWD
        HEAD(11)(6:8)=WD(1:3)
        ENCODE(8,200,LEN)ILEN
        HEAD(12)(6:8)=LEN(1:3)
C       TYPE *,HEAD
        IBRACKET=INDEX(FNAM,']')
        IPERIOD=INDEX(FNAM,'.')
        TNAME=FNAM(IPERIOD-9:IPERIOD-1)
        IBRACKET=IBRACKET
        IF(IPERIOD-10.LE.IBRACKET)THEN
        IZERO=ABS(IPERIOD-10)
        TNAME(1:IZERO)=' '
        ENDIF
        ILAST4=TNAME(6:9)
        IFIRST5=TNAME(1:5)

        HDR2ADR(1:8)=  'FN*X0000'
        HDR2ADR(11:18)='0000.IMG'
        HDR2ADR(4:8)=IFIRST5
        HDR2ADR(11:14)=ILAST4
        HDR2ADR(51:58)='SLREDALA'
        HDR2ADR(41:48)='LT000000'
```

[AVA.DISK]AVATODSK2

```
        HDR2ADR(31:38)='RT000000'
        HDR2ADR(21:28)='DT000000'
C       HDR2ADR(351:358)=MILISECONDS
        CALL TIME(TIMEA)
        HDR2ADR(43:44)=TIMEA(1:2)
        HDR2ADR(45:46)=TIMEA(4:5)
        HDR2ADR(47:48)=TIMEA(7:8)
C
C       LETS READ THE RANGE IRIG TIME FROM THE AVA FRAME BUFFER
C
        AVAACR='435'O
        X='1000'O
        Y=2
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(AVACHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
C       1D,%VAL(8),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1D,%VAL(8),%VAL(X),%VAL(Y),%VAL(1),%VAL(AVAACR))
C       IF(AVACSR.EQ.0)AVACSR=1
        DO I=2,4
        D(I)=NOT(D(I))
        ENDDO
        HMS=IAND(ISHFT(D(2),-8),'F'X)
        TMS=IAND(ISHFT(D(2),-4),'F'X)
        MS=IAND(D(2),'F'X)
        US=IAND(ISHFT(D(2),-12),'F'X)
        TS=IAND(D(3),7)
        UM=IAND(ISHFT(D(3),-3),'F'X)
        TM=IAND(ISHFT(D(3),-7),7)
        UH=IAND(ISHFT(D(3),-10),'F'X)
        TH=IAND(ISHFT(D(3),-14),3)
        UD=IAND(D(4),'F'X)
        TD=IAND(ISHFT(D(4),-4),'F'X)
        HD=IAND(ISHFT(D(4),-8),'F'X)
        HDR2ADR(33:33)=CHAR(TH+48)
        HDR2ADR(34:34)=CHAR(UH+48)
        HDR2ADR(35:35)=CHAR(TM+48)
        HDR2ADR(36:36)=CHAR(UM+48)
        HDR2ADR(37:37)=CHAR(TS+48)
        HDR2ADR(38:38)=CHAR(US+48)

C       HDR2ADR(351:358)=MILISECONDS
        HDR2ADR(279:279)=CHAR(HMS+48)
        HDR2ADR(280:280)=CHAR(TMS+48)
        HDR2ADR(281:281)=CHAR(MS +48)
C       WRITE(6,13)(D(I),I=2,4),HD,TD,UD,TH,UH,TM,UM,TS,US,
C       1HMS,TMS,MS
CC13    FORMAT(1X,3(1X,O6),5X,3Z1,':',1X,2Z1,':',Z1,Z1,':',2Z1,
C       1':',3Z1)
C       HDR2ADR(33:38)=HDR2ADR(43:48)
C       HDR2ADR(23:28)=HDR2ADR(43:48)
        HDR2ADR(23:28)=HEAD(2)(1:2)//HEAD(2)(4:5)//HEAD(2)(7:8)

        CALL UNCNVRT(%VAL(HDR2ADR),HDR2LEN,HDR2ADR)
C       TYPE*,'HDR2',HDR2ADR(1:HDR2LEN)
C       TYPE*,' WRITING ',FNAM(1:40)
```

[AVA.DISK]AVATODSK2

```
        IHD2=HDR2LEN    !AUTOWRTSB ROUTINE NEEDS THIS DEFINED THROUGH AUTOIMG.CMN
        ISTATUS=AUTOWRTSB(1,1,ILEN,IWD,IMAGE,%VAL(HDR2ADR))
        IF(.NOT.ISTATUS)TYPE *,'ERROR IN AUTOWRTSB IMAGE TO DISK'
        RETURN
        END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT,IOLINE)
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(NCOL+1,NROW),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+2
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END

        SUBROUTINE AVAREAD(IMAGE,AVACHAN)
        EXTERNAL IO$READVBLK
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INTEGER*2 IMAGE(NCOL+1,NROW)
        INTEGER AVACHAN,SYS$QIOW,AVACSR,AVAACR,X,Y
        INTEGER*2 INPUT(1536Ø),IOSB(4)
        BYTE BINPUT(3Ø72Ø)
        EQUIVALENCE(BINPUT,INPUT)
        AVACSR=Ø
        AVAACR='415'O
        Y=6
        X=Ø
        ICOUNT=Ø
        IADDR=1
        IOLINE=1
1       ISTATUS=SYS$QIOW(%VAL(1),%VAL(AVACHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
        IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (6,54)BINPUT
54      FORMAT(1X,16(1X,O3))
```

```
        NUMB=30720
        CALL BUFFCNVT(NUMB,BINPUT,IMAGE,IOLINE)
        Y=Y+30
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIOW(%VAL(1),%VAL(AVACHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,IMAGE,IOLINE)
        IOLINE=2
        Y='206'O
        X=0
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        ICOUNT=0
C       ISTATUS=SYS$QIOW(%VAL(1),%VAL(AVACHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,IMAGE,IOLINE)
        RETURN
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
         END
```

APPENDIX B

[AVA.MAXDISK]MTODSK3

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE CURRENT AVA FRAME FIELDS 1,2 AND WRITES
C       THEM TO DISK$IMAGES:[AVA]IMAGES.DAT AS THEY ARE PRESENTED BY PLAYING
C       THE HBR-3ØØØ BACK AT 32X1.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLSI
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 X,Y,YA(4),SYS$DASSGN
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(65536)
C       BYTE BINPUT(32768)
        BYTE BINPUT(131Ø72)
        INTEGER AVACSR,AVAACR,SYS$LKWSET,INLOCK(2),IOLOCK(2)
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF
        CHARACTER*6Ø TITLE,FNAME*6Ø
        CHARACTER*6Ø NAME
        INTEGER*2 BUFFERL,DEVCODE
        INTEGER SYS$GETDVI,DVI$FREEBLOCKS
        INTEGER IFREE
        INTEGER BUFFERA,ZERO
        COMMON/PRACHAN/IDISK
        COMMON/ITEMLIST/BUFFERL,DEVCODE,BUFFERA,ZERO
        COMMON/AVACHAN/ITCHAN
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        DATA YA/6,'2Ø6'O,'4Ø6'O,'6Ø6'O/
        DATA DVI$FREEBLOCKS/'ØØØØØØ2A'X/,ZERO/Ø/
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
        1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
        2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
        DATA ISETUP2/'64777'O,'44ØØØ'O/
        DATA ISETUP3/'64776'O,'44ØØØ'O/
        I = SYS$ASSIGN('GRAØ',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
```

```
[AVA.MAXDISK]MTODSK3

        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
        NAME='DISK$AVA:'
        BUFFERL=4
        DEVCODE=DVI$FREEBLOCKS
        BUFFERA=%LOC(IFREE)
C       RETURNLA=%LOC(RETURNL)
        ISTATUS=SYS$GETDVI(%VAL(3),,NAME,BUFFERL,,,,)

        IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN GETDVI'
        ISTATUS=SYS$WAITFR(%VAL(3))
        TYPE *,'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
        MAXIMAGES=IFREE/513
        TYPE *,'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C       MAXIMAGES=30    !THIS IS FOR DEGUG ONLY
C
        NIMAGES=MAXIMAGES
7775    INSZ=NIMAGES*480
        FNAME='DISK$AVA:[AVA]IMAGES.DAT'
        TYPE *,'OPENING',FNAME
        OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
        1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
        2RECORDTYPE='FIXED',RECORDSIZE=4096,ERR=777)
        GO TO 776
777     NIMAGES=NIMAGES-10
        IF(NIMAGES.LT.0)STOP 'NIMAGES LESS THAN ZERO!!!!!'
        GO TO 7775
776     TYPE*,'THE ACTUAL NUMBER OF IMAGES TO BE WRITTEN=',NIMAGES
        ISTATUS=SYS$ASSIGN('AVA0',AVACHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        INLOCK(1)=%LOC(BINPUT(1))
        INLOCK(2)=%LOC(BINPUT(131072))
        K=SYS$LKWSET(INLOCK,IOLOCK,)
        TYPE *,' INLOCK(1)= ',INLOCK(1),' INLOCK(2)= ',INLOCK(2)
        TYPE *,' IOLOCK(1)= ',IOLOCK(1),' IOLOCK(2)= ',IOLOCK(2)
        IF(.NOT.K)TYPE *,' UNABLE TO LOCK BUF'
        AVAACR='415'O
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BUF(1),%VAL(28),,,,)
        IEVFO=4
        IEVFO1=5
        IMAGEN=1
        IBLOCK=1
        AVACSR=0
        CALL TIMRB
        X=0
10      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.NE.0)GO TO 10
        ICURR=IFIELD
C       TYPE *,'ICURR=',ICURR
11      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.EQ.ICURR)GO TO 11
        ISTOREFIELD=ICURR       !CURRENT FIELD TO PUT ON DISK
```

```
[AVA.MAXDISK]MTODSK3

        ICURR=IFIELD             !CURRENT FIELD BE LOADED INTO THE AVA
C       TYPE *,'ICURR=',ICURR,'ISTOREFIELD=',ISTOREFIELD
C       GO TO 11
        ICOUNT=Ø
1       Y=YA(ISTOREFIELD+1)
        ISTATUS=SYS$WAITFR(%VAL(IEVFO1))
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø) AVACSR=1
        ISTATUS=SYS$QIO(%VAL(IEVFO1),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
2       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(32679),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(32679),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
3       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
4       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(983Ø5),%VAL(24576),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(983Ø5),%VAL(24576),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+48
C       IF(AVACSR.EQ.Ø)AVACSR=1

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       ICOUNT=ICOUNT+1
C       IF(ICOUNT.NE.4)GO TO 1
C       CALL FIELD(IFIELD,AVACSR)
C       IF(IFIELD.EQ.ICURR)GO TO 11
```

```
[AVA.MAXDISK]MTODSK3

C       IF(IFIELD.NE.IAND(ICURR+1,3))THEN
C       TYPE*,'FATAL ERROR....****....I/O TO SLOW'
C       TYPE *,'BLOCK NUMBER=',IBLOCK
C       TYPE *,'ICURR=',ICURR,' IFIELD=',IFIELD
C       CALL TIMRE
C       CALL HEADER(TITLE)
C       ISTATUS=SYS$DASSGN(%VAL(IDISK))
C       CLOSE(UNIT=3Ø)
C
C       STOP
C       ENDIF
C       ICURR=IAND(ICURR+1,3)
        IF(ICOUNT.EQ.Ø)THEN
        ICOUNT=1
        ISTOREFIELD=1
        GO TO 1
        ENDIF
C       TYPE*,IMAGEN/2,NIMAGES
        IF(IMAGEN.GE.NIMAGES)THEN
        TYPE *,IMAGEN, 'IMAGES WRITTEN TO DISK IN IMAGES.DAT'
        CALL TIMRE
        CALL HEADER(TITLE)
        ISTATUS=SYS$DASSGN(%VAL(IDISK))
        CLOSE(UNIT=3Ø)
        STOP 'I/O COMPLETE.............'
        ENDIF
        IMAGEN=IMAGEN+1
        GO TO 1Ø
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
```

[AVA.MAXDISK]MTODSK3

```
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END

        SUBROUTINE FIELD(IFIELD,AVACSR)
        INTEGER AVACSR
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
        INTEGER*2 INPUT,OUTPUT,INIT(4)
        CHARACTER *8Ø MSGBUF
        COMMON/AVACHAN/ITCHAN
        DATA IFIRST/1/
        ISAVE=AVACSR
        IF(IFIRST)THEN
        AVACSR='4ØØØ'O  !SET MEMORY WINDOW ENABLE AND INITIALIZE AVA
        IFIRST=Ø
        ELSE
        AVACSR='4ØØ1'O
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT,%VAL(2),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(IAVAACR))
C       IF(AVACSR.EQ.'4ØØØ'O)AVACSR='4ØØ1'O
C       IF(ISTATUS)     GO TO 5Ø1
C       TYPE *,' ERROR IN QIOW CALL'
C       ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C       MSGBUF=' '
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'I/O STATUS:',MSGBUF
5Ø1     AVACSR=ISAVE
        IFIELD=IAND(OUTPUT,3)
        RETURN
        END
```

APPENDIX C

[AVA.MAXDISK]MTODSK3A

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE CURRENT AVA FRAME FIELDS Ø,1 AND WRITES
C       THEM TO DISK$AVA:[AVA]IMAGES.DAT AS THEY ARE PRESENTED BY PLAYING
C       THE HBR-3ØØØ BACK AT 32X1.
C
C       THIS PROGRAM DIFFERS FROM MTODSK3 IN THAT THE IRIG TIME IN THE FRAME
C       BUFFER IS READ FOR EACH IMAGE AND STORED IN A BUFFER.
C       THE BUFFER IS OUTPUT AFTER ALL IMAGES HAVE BEEN TRANSFERED TO A FILE
C       NAMED DISK$AVA:[AVA]IRIGS.DAT. THESE IRIGS WILL CORRESPOND ONE TO ONE
C       WITH THE IMAGES IN IMAGES.DAT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLS1
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4),D(4)
        REAL*8 QD,IRIG(1ØØØ)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 X,Y,YA(4),SYS$DASSGN
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(65536)
        INTEGER*2 US,TS,UM,TM
        INTEGER*2 UH,TH,UD,TD,HD
        INTEGER*2 MS,HMS,TMS
C       BYTE BINPUT(32768)
        BYTE BINPUT(131Ø72)
        INTEGER AVACSR,AVAACR,SYS$LKWSET,INLOCK(2),IOLOCK(2)
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF
        CHARACTER*6Ø TITLE,FNAME*6Ø
        CHARACTER*6Ø NAME
        INTEGER*2 BUFFERL,DEVCODE
        INTEGER SYS$GETDVI,DVI$FREEBLOCKS
        INTEGER IFREE
        INTEGER BUFFERA,ZERO
        COMMON/PRACHAN/IDISK
        COMMON/ITEMLIST/BUFFERL,DEVCODE,BUFFERA,ZERO
        COMMON/AVACHAN/ITCHAN
        EQUIVALENCE(BUF(1),ISETUP(1))
```

```
[AVA.MAXDISK]MTODSK3A

          EQUIVALENCE(BINPUT,INPUT)
          EQUIVALENCE(D,QD)
          DATA YA/6,'2Ø6'O,'4Ø6'O,'6Ø6'O/
          DATA DVI$FREEBLOCKS/'ØØØØØØ2A'X/,ZERO/Ø/
          DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
          1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
          2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
          DATA ISETUP2/'64777'O,'44ØØØ'O/
          DATA ISETUP3/'64776'O,'44ØØØ'O/
          I = SYS$ASSIGN('GRAØ',CHAN,,)
          IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
          ISTATUS=SYS$ASSIGN('AVAØ',ITCHAN,,)
          IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
          TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
          NAME='DISK$AVA:'
          BUFFERL=4
          DEVCODE=DVI$FREEBLOCKS
          BUFFERA=%LOC(IFREE)
C         RETURNLA=%LOC(RETURNL)
          ISTATUS=SYS$GETDVI(%VAL(3),,NAME,BUFFERL,,,,)

          IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN GETDVI'
          ISTATUS=SYS$WAITFR(%VAL(3))
          TYPE *,'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
          MAXIMAGES=IFREE/481
          TYPE *,'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C         MAXIMAGES=5      !THIS IS FOR DEGUG ONLY
C
          NIMAGES=MAXIMAGES-2
7775      INSZ=NIMAGES*48Ø
          FNAME='DISK$AVA:[AVA]IMAGES.DAT'
          TYPE *,'OPENING',FNAME
          OPEN(UNIT=3Ø,NAME=FNAME,TYPE='UNKNOWN',
          1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
          2RECORDTYPE='FIXED',RECORDSIZE=4Ø96,ERR=777)
          GO TO 776
777       NIMAGES=NIMAGES-1Ø
          IF(NIMAGES.LT.Ø)STOP 'NIMAGES LESS THAN ZERO!!!!!'
          GO TO 7775
776       TYPE*,'THE ACTUAL NUMBER OF IMAGES TO BE WRITTEN=',NIMAGES
          ISTATUS=SYS$ASSIGN('AVAØ',AVACHAN,,)
          IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
          INLOCK(1)=%LOC(BINPUT(1))
          INLOCK(2)=%LOC(BINPUT(131Ø72))
          K=SYS$LKWSET(INLOCK,IOLOCK,)
          TYPE *,' INLOCK(1)= ',INLOCK(1),' INLOCK(2)= ',INLOCK(2)
          TYPE *,' IOLOCK(1)= ',IOLOCK(1),' IOLOCK(2)= ',IOLOCK(2)
          IF(.NOT.K)TYPE *,' UNABLE TO LOCK BUF'
C         K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C         1BUF(1),%VAL(28),,,,)
          IEVFO=4
          IMAGEN=1
          IBLOCK=1
          AVACSR=Ø
```

```
[AVA.MAXDISK]MTODSK3A

        CALL TIMRB
        X=Ø
        AVAACR='415'O
1Ø      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.NE.Ø)GO TO 1Ø
        ICURR=IFIELD
C       TYPE *,'ICURR=',ICURR
C
C       GET IRIG
C
C       AVACSR=Ø
        AVAACR='435'O
        X='1ØØØ'O
        Y=2
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1D,%VAL(8),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IRIG(IMAGEN)=QD
C
C       GO GET SECOND FIELD IN THIS FRAME
C
        AVAACR='415'O
        X=Ø
11      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.EQ.ICURR)GO TO 11
        ISTOREFIELD=ICURR        !CURRENT FIELD TO PUT ON DISK
        ICURR=IFIELD             !CURRENT FIELD BE LOADED INTO THE AVA
C       TYPE *,'ICURR=',ICURR,'ISTOREFIELD=',ISTOREFIELD
C       GO TO 11
        ICOUNT=Ø
1       Y=YA(ISTOREFIELD+1)
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø) AVACSR=1
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
        ISADR=32769
2       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(ISADR),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(ISADR),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
3       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(IBLOCK),,,)
```

```
[AVA.MAXDISK]MTODSK3A

        IBLOCK=IBLOCK+64
        Y=Y+32
4       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(98305),%VAL(24576),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(98305),%VAL(24576),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+48
        IF(ICOUNT.EQ.0)THEN
        ICOUNT=1
        ISTOREFIELD=1
        GO TO 1
        ENDIF
C       TYPE*,IMAGEN/2,NIMAGES
        IF(IMAGEN.GE.NIMAGES)THEN
        TYPE *,IMAGEN, 'IMAGES WRITTEN TO DISK IN IMAGES.DAT'
        CALL TIMRE
        CALL HEADER(TITLE)
        ISTATUS=SYS$DASSGN(%VAL(IDISK))
        CLOSE(UNIT=30)
C
C       PROCESS IRIG BUFFER
C
        OPEN(UNIT=9,NAME='DISK$AVA:[AVA]IRIGS.DAT',STATUS='NEW')
        DO IRIGX=1,IMAGEN
        QD=IRIG(IRIGX)
        DO I=2,4
        D(I)=NOT(D(I))
        ENDDO
        HMS=IAND(ISHFT(D(2),-8),'F'X)
        TMS=IAND(ISHFT(D(2),-4),'F'X)
        MS=IAND(D(2),'F'X)
        US=IAND(ISHFT(D(2),-12),'F'X)
        TS=IAND(D(3),7)
        UM=IAND(ISHFT(D(3),-3),'F'X)
        TM=IAND(ISHFT(D(3),-7),7)
        UH=IAND(ISHFT(D(3),-10),'F'X)
        TH=IAND(ISHFT(D(3),-14),3)
        UD=IAND(D(4),'F'X)
        TD=IAND(ISHFT(D(4),-4),'F'X)
        HD=IAND(ISHFT(D(4),-8),'F'X)

        WRITE(9,13)HD,TD,UD,TH,UH,TM,UM,TS,US,HMS,TMS,MS
13      FORMAT(3Z1,':',1X,2Z1,':',Z1,Z1,':',2Z1,':',3Z1)
        ENDDO
        STOP 'I/O COMPLETE.............'
        ENDIF
        IMAGEN=IMAGEN+1
        GO TO 10
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END

        SUBROUTINE FIELD(IFIELD,AVACSR)
        INTEGER AVACSR
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
        INTEGER*2 INPUT,OUTPUT,INIT(4)
        CHARACTER *8Ø MSGBUF
        COMMON/AVACHAN/ITCHAN
        DATA IFIRST/1/
        ISAVE=AVACSR
        IF(IFIRST)THEN
        AVACSR='4ØØØ'O  !SET MEMORY WINDOW ENABLE AND INITIALIZE AVA
        IFIRST=Ø
        ELSE
        AVACSR='4ØØ1'O
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT,%VAL(2),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(IAVAACR))
C       IF(AVACSR.EQ.'4000'O)AVACSR='4001'O
C       IF(ISTATUS)     GO TO 501
C       TYPE *,' ERROR IN QIOW CALL'
C       ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C       MSGBUF=' '
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'I/O STATUS:',MSGBUF
501     AVACSR=ISAVE
        IFIELD=IAND(OUTPUT,3)
        RETURN
        END
```

APPENDIX D

[AVA.MAXDISK]MTODSK4

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE CURRENT AVA FRAME FIELD AND WRITES IT
C       IT TO DISK$IMAGES:[AVA]IMAGES.DAT. THE NEXT FIELD IS THEN WRITTEN
C       TO DISK ALSO IF THERE IS ENOUGH TIME ELSE A FIELD IS SKIPPED AND THE
C       PROCESS CONTINUES.
C
C       THIS PROGRAM IS DIFFERENT FROM MTODSK3 IN THAT IT TRANSFERS THE
C       CURRENT FIELD AND FOLLOWING FIELDS WHERE AS MTODSK3 ALWAYS TRANSFERS
C       FIELD Ø THEN FIELD 1 AND DOES NOT ATTEMPT TO TRANSFER FIELDS 2 OR 3.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLS1
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 X,Y,YA(4),SYS$DASSGN
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(65536)
C       BYTE BINPUT(32768)
        BYTE BINPUT(131Ø72)
        INTEGER AVACSR,AVAACR,SYS$LKWSET,INLOCK(2),IOLOCK(2)
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF
        CHARACTER*6Ø TITLE,FNAME*6Ø
        CHARACTER*6Ø NAME
        INTEGER*2 BUFFERL,DEVCODE
        INTEGER SYS$GETDVI,DVI$FREEBLOCKS
        INTEGER IFREE
        INTEGER BUFFERA,ZERO
        COMMON/PRACHAN/IDISK
        COMMON/ITEMLIST/BUFFERL,DEVCODE,BUFFERA,ZERO
        COMMON/AVACHAN/ITCHAN
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        DATA YA/6,'2Ø6'O,'4Ø6'O,'6Ø6'O/
        DATA DVI$FREEBLOCKS/'ØØØØØØ2A'X/,ZERO/Ø/
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
        1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
```

[AVA.MAXDISK]MTODSK4

```
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64776'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
        NAME='DISK$AVA:'
        BUFFERL=4
        DEVCODE=DVI$FREEBLOCKS
        BUFFERA=%LOC(IFREE)
C       RETURNLA=%LOC(RETURNL)
        ISTATUS=SYS$GETDVI(%VAL(3),,NAME,BUFFERL,,,,)

        IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN GETDVI'
        ISTATUS=SYS$WAITFR(%VAL(3))
        TYPE *,'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
        MAXIMAGES=IFREE/513
        TYPE *,'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C       MAXIMAGES=30    !THIS IS FOR DEGUG ONLY
C
        NIMAGES=MAXIMAGES
7775    INSZ=NIMAGES*481
        FNAME='DISK$AVA:[AVA]IMAGES.DAT'
        TYPE *,'OPENING',FNAME
        OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
        1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
        2RECORDTYPE='FIXED',RECORDSIZE=4096,ERR=777)
        GO TO 776
777     NIMAGES=NIMAGES-10
        IF(NIMAGES.LT.0)STOP 'NIMAGES LESS THAN ZERO!!!!!'
        GO TO 7775
776     TYPE*,'THE ACTUAL NUMBER OF IMAGES TO BE WRITTEN=',NIMAGES
        ISTATUS=SYS$ASSIGN('AVA0',AVACHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        INLOCK(1)=%LOC(BINPUT(1))
        INLOCK(2)=%LOC(BINPUT(131072))
        K=SYS$LKWSET(INLOCK,IOLOCK,)
        TYPE *,' INLOCK(1)= ',INLOCK(1),' INLOCK(2)= ',INLOCK(2)
        TYPE *,' IOLOCK(1)= ',IOLOCK(1),' IOLOCK(2)= ',IOLOCK(2)
        IF(.NOT.K)TYPE *,' UNABLE TO LOCK BUF'
        AVAACR='415'O
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BUF(1),%VAL(28),,,,)
        IEVFO=4
        IMAGEN=1
        IBLOCK=1
        AVACSR=0
        CALL TIMRB
        X=0
        CALL FIELD(IFIELD,AVACSR)
        ICURR=IFIELD
        GO TO 11
```

[AVA.MAXDISK]MTODSK4

```
1Ø      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.NE.ICURR)GO TO 1Ø
        ICURR=IFIELD
C       TYPE *,'ICURR=',ICURR
11      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.EQ.ICURR)GO TO 11
        ISTOREFIELD=ICURR        !CURRENT FIELD TO PUT ON DISK
        ICURR=IFIELD             !CURRENT FIELD BE LOADED INTO THE AVA
C       TYPE *,'ICURR=',ICURR,'ISTOREFIELD=',ISTOREFIELD
C       GO TO 11
        ICOUNT=Ø
1       ISTOREF=ISTOREFIELD+1
        Y=YA(ISTOREF)
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø) AVACSR=1
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
2       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(32679),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(32679),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
3       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
4       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(983Ø5),%VAL(24576),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1BINPUT(983Ø5),%VAL(24576),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+48
        IF(ICOUNT.EQ.Ø)THEN
                ICOUNT=1
                ISTOREFIELD=ISTOREFIELD+1
                IF(ISTOREFIELD.GT.3)ISTOREFIELD=Ø
                GO TO 1
        ENDIF
                IF(IMAGEN.GE.NIMAGES)THEN
                TYPE *,'I/O COMPLETE..........................'
                TYPE *,IMAGEN,' IMAGES WRITTEN TO DISK$AVA:[AVA]IMAGES.DAT'
                CALL TIMRE
```

```
[AVA.MAXDISK]MTODSK4

                    CALL HEADER(TITLE)
                    ISTATUS=SYS$DASSGN(%VAL(IDISK))
                    CLOSE(UNIT=30)
                    STOP 'IMAGE WRITTEN TO DISK'
                    ENDIF
          IMAGEN=IMAGEN+1
          GO TO 10
57        CONTINUE
          ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
          TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
          TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
          IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
          TYPE *,'QIO PARAMETER STATUS:',MSGBUF
          MSGBUF=' '
          ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
          IF(.NOT.ISTATUS)TYPE *,'ERROR IN CALL TO $GETMSG'
          TYPE *,'I/O STATUS:',MSGBUF
          STOP
C11       FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C         K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C         1ISETUP3,%VAL(4),,,,)
           END
          SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
          BYTE BINPUT(1),BYTE(2)
          INTEGER*2 OUT(513,1),BYTES,SLU
          EQUIVALENCE(BYTES,BYTE)
          DATA SLU/'34011'O/
          I=0
          IOLINE=1
          DO 100 IX=1,NUMB
          I=I+1
          IF(I.EQ.512)THEN
          BYTE(1)=BINPUT(IX)
          OUT(I,IOLINE)=BYTES
C         WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
          OUT(I+1,IOLINE)=SLU
C         WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
          I=0
          IOLINE=IOLINE+1
          GO TO 100
          ENDIF
          BYTE(1)=BINPUT(IX)
          OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C         WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34        FORMAT(1X,I3,1X,I3,2X,O6)
100       CONTINUE
          RETURN
          END

          SUBROUTINE FIELD(IFIELD,AVACSR)
          INTEGER AVACSR
          EXTERNAL IO$WRITEVBLK,IO$READVBLK
          INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
          INTEGER SYS$GETMSG
          INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
```

[AVA.MAXDISK]MTODSK4

```
        INTEGER*2 INPUT,OUTPUT,INIT(4)
        CHARACTER *8Ø MSGBUF
        COMMON/AVACHAN/ITCHAN
        DATA IFIRST/1/
        ISAVE=AVACSR
        IF(IFIRST)THEN
        AVACSR='4ØØØ'O  !SET MEMORY WINDOW ENABLE AND INITIALIZE AVA
        IFIRST=Ø
        ELSE
        AVACSR='4ØØ1'O
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT,%VAL(2),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(IAVAACR))
C       IF(AVACSR.EQ.'4ØØØ'O)AVACSR='4ØØ1'O
C       IF(ISTATUS)     GO TO 5Ø1
C       TYPE *,' ERROR IN QIOW CALL'
C       ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C       MSGBUF=' '
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'I/O STATUS:',MSGBUF
5Ø1     AVACSR=ISAVE
        IFIELD=IAND(OUTPUT,3)
        RETURN
        END
```

APPENDIX E

[AVA.MAXDISK]MDSKTOGRN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND DISPLAYS
C       THE IMAGE ON THE GRINNELL.
C
C       THIS IS THE COMPACT IMAGE FORMAT USED WHEN TRYING TO OUTPUT THE AVA
C       IMAGE AS FAST AS POSSIBLE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLSI
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2),IIMAGEB(4)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        BYTE BINPUT(32768)
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF
        CHARACTER*6Ø TITLE,FNAME
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        COMMON/PRACHAN/IDISK
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
        1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
        2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
        DATA ISETUP2/'64777'O,'44ØØØ'O/
        DATA ISETUP3/'64776'O,'44ØØØ'O/
        I = SYS$ASSIGN('GRAØ',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=Ø
        AVAACR='415'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
```

```
[AVA.MAXDISK]MDSKTOGRN

        TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
        NIMAGES=1
        INSZ=NIMAGES*513
        FNAME='DISK$AVA:[AVA]IMAGES.DAT'
        OPEN(UNIT=3Ø,NAME=FNAME,TYPE='UNKNOWN',
        1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
        2RECORDTYPE='FIXED',RECORDSIZE=4Ø96)
        IEVFO=4
        Y=6
        X=Ø
        TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
        ACCEPT*,IBLOCK
        IF(IBLOCK.EQ.1)THEN
        IBLOCK=1
        ISTARTI=1
        ELSE
        ISTARTI=IBLOCK
        IBLOCK=1+(24Ø*(IBLOCK-1))
        ENDIF
        TYPE*,'ENTER IMAGE INCREMENT'
        ACCEPT*,IBLOCKI
        ICOUNT=Ø
C       CALL TIMRB
        TYPE *,'ENTER NUMBER OF IMAGES TO DISPLAY'
        ACCEPT*,IFIELDS
C       TYPE *,IFIELDS
        IFIELDS=IFIELDS*8
1       IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        NBYTES=24576
        ELSE
        NBYTES=32768
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(NBYTES),%VAL(IBLOCK),,,)
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IBLOCK=IBLOCK+48
        NUMB=24576
        ELSE
        NUMB=32768
        IBLOCK=IBLOCK+64
        ENDIF
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IGBYTES=24624
        ELSE
        IGBYTES=32832
        ENDIF

        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(IGBYTES),,,,)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(IGBYTES+1),%VAL(IGBYTES),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        IIMAGEB(1)=27
        IIMAGEB(2)=89
        IIMAGEB(3)=55
        IIMAGEB(4)=4Ø
        WRITE(6,77)IIMAGEB,ISTARTI
77      FORMAT(1H+,4A1,'IMAGE NUMBER',I5, ' DISPLAYED ON THE GRINNELL NOW.')
        ICOUNT=Ø
        ISTARTI=ISTARTI+IBLOCKI
        IBLOCK=IBLOCK+(24Ø*(IBLOCKI-1))
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP2(1),%VAL(4),,,,)
C       CALL TIMRE
C       CALL HEADER(TITLE)
C       STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
        Y=6
        ENDIF
        IFIELDS=IFIELDS-1
        IF(IFIELDS.EQ.Ø)STOP
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
```

```
[AVA.MAXDISK]MDSKTOGRN

        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX F

[AVA.MAXDISK]MDTOGRN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C        THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND DISPLAYS
C        THE IMAGE ON THE GRINNELL A FRAME AT A TIME.
C
C        THIS IS THE COMPACT IMAGE FORMAT USED WHEN TRYING TO OUTPUT THE AVA
C        IMAGE AS FAST AS POSSIBLE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLS1
         INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
         INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
         INTEGER SYS$GETMSG,MSGLEN,ISTATUS
         INTEGER*2 OUT(513,480),X,Y
         BYTE BOUT(492480),BYTE(2),IIMAGEB(4)
         INTEGER*2 BYTES
         INTEGER*2 OUTPUT,INIT(4)
         INTEGER*2 INPUT(16384)
         BYTE BINPUT(32768),BLINES(512,64)
         INTEGER AVACSR,AVAACR
         INTEGER*2 ISETUP2(2),ISETUP3(2)
         CHARACTER *80 MSGBUF
         CHARACTER*60 TITLE,FNAME
         EQUIVALENCE(BUF(1),ISETUP(1))
         EQUIVALENCE(BINPUT,INPUT)
         EQUIVALENCE(BOUT,OUT),(BYTE,BYTES),(BINPUT,BLINES)
         COMMON/PRACHAN/IDISK
         DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
         1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
         2 '50001'O,'70777'O,'54000'O/
         DATA ISETUP2/'64777'O,'44000'O/
         DATA ISETUP3/'64776'O,'44000'O/
         I = SYS$ASSIGN('GRA0',CHAN,,)
         IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
C        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
C        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
         AVACSR=0
         AVAACR='415'O
         K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
         1BUF(1),%VAL(28),,,,)
```

```
[AVA.MAXDISK]MDTOGRN

        TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
        NIMAGES=1
        INSZ=NIMAGES*513
        FNAME='DISK$AVA:[AVA]IMAGES.DAT'
        OPEN(UNIT=3Ø,NAME=FNAME,TYPE='UNKNOWN',
        1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
        2RECORDTYPE='FIXED',RECORDSIZE=4Ø96)
        IEVFO=4
        Y=6
        X=Ø
        TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
        ACCEPT*,IBLOCK
        IF(IBLOCK.EQ.1)THEN
        IBLOCK=1
        ISTARTI=1
        ELSE
        ISTARTI=IBLOCK
        IBLOCK=1+(48Ø*(IBLOCK-1))
C       48Ø BLOCKS CONTAINS ONE FULL IMAGE OR TWO FIELDS
        ENDIF
        TYPE*,'ENTER IMAGE INCREMENT'
        ACCEPT*,IBLOCKI
        ICOUNT=Ø
C       CALL TIMRB
        TYPE *,'ENTER NUMBER OF IMAGES TO DISPLAY'
        ACCEPT*,IFIELDS
C       TYPE *,IFIELDS
        IFIELDS=IFIELDS*8
1       IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        NBYTES=24576
        ELSE
        NBYTES=32768
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(IDISK),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(NBYTES),%VAL(IBLOCK),,,)
        IF(.NOT.ISTATUS)TYPE *,'QIO PARAMETER ERROR ON DISK READ'
        IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN DISK INPUT'
C       TYPE *,'IBLOCK=',IBLOCK
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IBLOCK=IBLOCK+48
        NUMB=24576
        ELSE
        NUMB=32768
        IBLOCK=IBLOCK+64
        ENDIF
C       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))
```

[AVA.MAXDISK]MDTOGRN

```
C       IF (ICOUNT.EQ.Ø)THEN
C       DO I=2,12
C       WRITE (6,155)(BLINES(J,I),J=8,16)
C       ENDDO
C       ENDIF
155     FORMAT(1X,1Ø(1X,O3))
189     CALL BUFFCNVT(NUMB,BINPUT,OUT,ICOUNT)
C       TYPE *,(OUT(I,1),I=1,2Ø)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        IF(ICOUNT.EQ.7)THEN
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP2(1),%VAL(4),,,,)
        DO IQIO=1,7
        IADDR=1+(65534*(IQIO-1))
        ISTATUS = SYS$QIOW(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(IADDR),%VAL(65534),,,,)
        ENDDO
        IADDR=1+(65534*(IQIO-1))
        ISTATUS = SYS$QIOW(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(IADDR),%VAL(33742),,,,)
        ENDIF
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        IIMAGEB(1)=27
        IIMAGEB(2)=89
        IIMAGEB(3)=55
        IIMAGEB(4)=4Ø
        WRITE(6,77)IIMAGEB,ISTARTI
77      FORMAT(1H+,4A1,'IMAGE NUMBER',I5, ' DISPLAYED ON THE GRINNELL NOW.')
        ICOUNT=Ø
        ISTARTI=ISTARTI+IBLOCKI
        IBLOCK=IBLOCK+(48Ø*(IBLOCKI-1))
C       48Ø BLOCKS CONTAINS ONE FULL IMAGE OR TWO FIELDS
C       CALL TIMRE
C       CALL HEADER(TITLE)
C       STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
        Y=6
        ENDIF
        IFIELDS=IFIELDS-1
        IF(IFIELDS.EQ.Ø)STOP
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
```

[AVA.MAXDISK]MDTOGRN

```
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT,ICOUNT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IF(ICOUNT.EQ.Ø)IOLINE=1
        IF(ICOUNT.EQ.1)IOLINE=129
        IF(ICOUNT.EQ.2)IOLINE=257
        IF(ICOUNT.EQ.3)IOLINE=385
        IF(ICOUNT.EQ.4)IOLINE=2
        IF(ICOUNT.EQ.5)IOLINE=13Ø
        IF(ICOUNT.EQ.6)IOLINE=258
        IF(ICOUNT.EQ.7)IOLINE=386

C       TYPE *,'IOLINE=',IOLINE,ICOUNT
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+2
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX G

[AVA.MAXDISK]FIELDSGRN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND DISPLAYS
C       THE FIELDS ON THE GRINNELL ONE AT A TIME.
C
C       THIS IS THE COMPACT IMAGE FORMAT USED WHEN TRYING TO OUTPUT THE AVA
C       IMAGE AS FAST AS POSSIBLE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLSI
        INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2),IIMAGEB(4)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        BYTE BINPUT(32768)
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *80 MSGBUF
        CHARACTER*60 TITLE,FNAME
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        COMMON/PRACHAN/IDISK
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24071'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64777'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=0
        AVAACR='415'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
```

```
[AVA.MAXDISK]FIELDSGRN

        TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
        NIMAGES=1
        INSZ=NIMAGES*513
        FNAME='DISK$AVA:[AVA]IMAGES.DAT'
        OPEN(UNIT=3Ø,NAME=FNAME,TYPE='UNKNOWN',
        1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
        2RECORDTYPE='FIXED',RECORDSIZE=4Ø96)
        IEVFO=4
        Y=6
        X=Ø
        TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
        ACCEPT*,IBLOCK
        IF(IBLOCK.EQ.1)THEN
        IBLOCK=1
        ISTARTI=1
        ELSE
        ISTARTI=IBLOCK
        IBLOCK=1+(24Ø*(IBLOCK-1))
        ENDIF
        TYPE*,'ENTER FIELD INCREMENT'
        ACCEPT*,IBLOCKI
        ICOUNT=Ø
C       CALL TIMRB
        TYPE *,'ENTER NUMBER OF IMAGES TO DISPLAY'
        ACCEPT*,IFIELDS
C       TYPE *,IFIELDS
        IFIELDS=IFIELDS*8
1       IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        NBYTES=24576
        ELSE
        NBYTES=32768
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(NBYTES),%VAL(IBLOCK),,,)
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IBLOCK=IBLOCK+48
        NUMB=24576
        ELSE
        NUMB=32768
        IBLOCK=IBLOCK+64
        ENDIF
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
```

```
         IGBYTES=24624
         ELSE
         IGBYTES=32832
         ENDIF

         ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
         1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
         1BOUT(1),%VAL(IGBYTES),,,,)
         ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
         1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
         1BOUT(IGBYTES+1),%VAL(IGBYTES),,,,)
C        IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
         Y=Y+32
         ICOUNT=ICOUNT+1
         IF(ICOUNT.EQ.4)THEN
         IIMAGEB(1)=27
         IIMAGEB(2)=89
         IIMAGEB(3)=55
         IIMAGEB(4)=40
         WRITE(6,77)IIMAGEB,ISTARTI
         K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
         1ISETUP3(1),%VAL(4),,,,)
         ISTARTI=ISTARTI+IBLOCKI
         Y='206'O
         X=0
         ENDIF
         IF(ICOUNT.EQ.8)THEN
         IIMAGEB(1)=27
         IIMAGEB(2)=89
         IIMAGEB(3)=55
         IIMAGEB(4)=40
         WRITE(6,77)IIMAGEB,ISTARTI
77       FORMAT(1H+,4A1,'FIELD NUMBER',I5, ' DISPLAYED ON THE GRINNELL NOW.')
         ICOUNT=0
         ISTARTI=ISTARTI+IBLOCKI
         IBLOCK=IBLOCK+(240*(IBLOCKI-1))
         K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
         1ISETUP3(1),%VAL(4),,,,)
C        CALL TIMRE
C        CALL HEADER(TITLE)
C        STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
         Y=6
         ENDIF
         IFIELDS=IFIELDS-1
         IF(IFIELDS.EQ.0)STOP
         GO TO 1
57       CONTINUE
         ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
         TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
         TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
         IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
         TYPE *,'QIO PARAMETER STATUS:',MSGBUF
         MSGBUF=' '
         ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
         IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
```

[AVA.MAXDISK]FIELDSGRN

```
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX H

[AVA.MAXDISK]MDSKTOFIL

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND GENERATES
C       NATO FORMATTED DISK FILES FOR AS MANY FILES AS SPECIFIED BY THE USER.
C
C       IMAGES.DAT IS THE COMPACT IMAGE FORMAT.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLS1,MITLS2
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*4 LIB$FREEVM,LIB$GETVM,SYS$DASSGN
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        BYTE BINPUT(32768)
        INTEGER AVACSR,AVAACR,OTS$CVTLTI
        INTEGER*2 ISETUP2(2),ISETUP3(2)
C        CHARACTER *8Ø MSGBUF,NAMNUM*4,DISKSPEC*14
        CHARACTER *8Ø MSGBUF,NAMNUM*4,DISKSPEC*22
        CHARACTER*6Ø TITLE,FNAME,FNAM2
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        COMMON/PRACHAN2/IDISK2
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
        1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
        2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
        DATA ISETUP2/'64777'O,'44ØØØ'O/
        DATA ISETUP3/'64776'O,'44ØØØ'O/
        DATA IFIRST/1/
```

[AVA.MAXDISK]MDSKTOFIL

```
        I=SYS$ASSIGN('TT',IVTC,,)
        IF(.NOT.I)TYPE *,'ERROR IN TT CHANNELL ASSIGN'
        I = SYS$ASSIGN('GRAØ',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=Ø
        AVAACR='415'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
        NIMAGES=1
        INSZ=NIMAGES*513
        FNAME='DISK$AVA:[AVA]IMAGES.DAT'
        OPEN(UNIT=3Ø,NAME=FNAME,TYPE='UNKNOWN',
        1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS2,
        2RECORDTYPE='FIXED',RECORDSIZE=4Ø96)
        IEVFO=4
        Y=6
        X=Ø
        TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
        ACCEPT*,IBLOCK
        IF(IBLOCK.EQ.1)THEN
        IBLOCK=1
        ELSE
        IBLOCK=1+(48Ø*(IBLOCK-1))
        ENDIF
        ICOUNT=Ø
C       CALL TIMRB
        TYPE *,'ENTER NUMBER IMAGES TO STORE'
        ACCEPT*,IFIELDS
        IFIELDS=IFIELDS*8          !CONVERT IMAGES TO TRANSFERS
1       IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        NBYTES=24576
        ELSE
        NBYTES=32768
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(IEVFO),%VAL(IDISK2),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(1),%VAL(NBYTES),%VAL(IBLOCK),,,)
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IBLOCK=IBLOCK+48
        NUMB=24576
        ELSE
        NUMB=32768
        IBLOCK=IBLOCK+64
        ENDIF
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
```

```
[AVA.MAXDISK]MDSKTOFIL

54      FORMAT(1X,16(1X,O3))

        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IGBYTES=24624
        ELSE
        IGBYTES=32832
        ENDIF

        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(IGBYTES),,,,)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(IGBYTES+1),%VAL(IGBYTES),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        IF(IFIRST)THEN
C       TYPE *,'ENTER DISK NAME AND DIRECTORY. (DISK$IMAGES:[WILLIAMS])'
C       READ(5,6)DISKSPEC
        DISKSPEC='DISK$IMAGES:[WILLIAMS]'
C       DISKSPEC='DISK$AVA:[AVA]'
        TYPE *,'ENTER FIRST IMAGE FILE NAME. (WØØØ1ØØØØ.IMG)'
        READ(5,6)FNAM2
6       FORMAT(A)
C       FNAM='SIMSØØØØ1.IMG'
        NAMNUM=FNAM2(2:5)
        FNAM=DISKSPEC//FNAM2
        DECODE(4,1Ø1,NAMNUM)INAMNUM
1Ø1     FORMAT(I4)
        NCOL=512
        NROW=48Ø
        ILEN=NROW
        IWD=NCOL
C       IMGMAPC(3)=ILEN !LENGTH OF IMAGE
C       IMGMAPC(4)=IWD  !WIDTH  OF IMAGE
        IFIRST=Ø
        I=IWD
        NBYT=(I+1)*ILEN*2
        I=LIB$GETVM(NBYT,IMGADR)
        IF(.NOT.I)TYPE *,' ERROR IN VIRTUAL MEMORY ASSIGNMENT  1'
        I = LIB$GETVM(1ØØØØ,HDR2ADR)
        IF(.NOT. I) CALL ERRSTOP(I,'ERROR GETTING HDR2 VM','AVATODSK')
        ENDIF
        HEAD(8)='        1'       !ONE CHARACTER PER CHANNEL
        HDR2LEN=576
        CURRENTNUMFL=Ø
        CALL ADDHDR2(%VAL(HDR2ADR))
        CALL IMGTODISK(BINPUT,NUMB,%VAL(IMGADR),ICOUNT)
        IF(ICOUNT.EQ.7)THEN
C       FNAM='SIMSØØØØ1.IMG'
        INAMNUM=INAMNUM+1
        ISTATUS=OTS$CVTLTI(INAMNUM,NAMNUM,%VAL(4),%VAL(4),)
        IF(.NOT.ISTATUS)TYPE *,'CONVERSION ERROR IN FILE NAME'
        FNAM2(2:5)=NAMNUM
```

[AVA.MAXDISK]MDSKTOFIL

```
        FNAM=DISKSPEC//FNAM2
        ENDIF
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        ICOUNT=Ø
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP2(1),%VAL(4),,,,)
C       CALL TIMRE
C       CALL HEADER(TITLE)
C       STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
        Y=6
        ENDIF
        IFIELDS=IFIELDS-1
        IF(IFIELDS.EQ.Ø)STOP 'ALL IMAGES WRITTEN TO DISK'
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
```

[AVA.MAXDISK]MDSKTOFIL

```
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
100     CONTINUE
        RETURN
        END

        SUBROUTINE IMGTODISK(BINPUT,NUMB,IMAGE,ICOUNT)
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
        INTEGER*2 IMAGE(NCOL+1,NROW)
        INTEGER*4 IMGADR,SYS$ASSIGN,IMGADR2,SYS$GETMSG
        BYTE BINPUT(1)
        INDEX=1
        IF(ICOUNT.LE.3)THEN
        ISTART=1+(ICOUNT*128)
        IF(ICOUNT.LE.2)IEND=ISTART+126
        IF(ICOUNT.EQ.3)IEND=ISTART+94
        DO I=ISTART,IEND,2
        DO J=1,NCOL
        INPUT=BINPUT(INDEX)
        IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C       IMAGE(J,I)=BINPUT(INDEX)
        INDEX=INDEX+1
        ENDDO
        ENDDO
        ELSE
        ISTART=2+((ICOUNT-4)*128)
        IF(ICOUNT.LE.6)IEND=ISTART+126
        IF(ICOUNT.EQ.7)IEND=ISTART+94
        DO I=ISTART,IEND,2
        DO J=1,NCOL
        INPUT=BINPUT(INDEX)
        IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C       IMAGE(J,I)=BINPUT(INDEX)
        INDEX=INDEX+1
        ENDDO
        ENDDO
        ENDIF
        ILEN=NROW
        IWD=NCOL
C       TYPE*,'IWD AND ILEN BEFORE TODISK=',IWD,ILEN
        IF(ICOUNT.EQ.7) CALL TODISK(IMAGE,IWD,ILEN)

C       WRITE(6,1)ICOUNT
1       FORMAT(1X,I10)
        RETURN
        END
        SUBROUTINE TODISK(IMAGE,IWD,ILEN)
```

[AVA.MAXDISK]MDSKTOFIL

```
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]AUTOIMG.CMN/NOLIST'
         INTEGER*2 IMAGE(NCOL+1,NROW),HDR2LEN
         INTEGER*4 AUTOWRTSB
         CHARACTER*10000 HDR2ADR
         CHARACTER*3 MONTH,DAY,YEAR*2,WD*8,LEN*8,TIMEA*8
         CHARACTER*5 IFIRST5,ILAST4*4,TNAME*9
         HDR2LEN=HDR2LEN
C        TYPE *,'HDR2LEN',HDR2LEN
C        TYPE *,'HDR2LEN ',HDR2LEN
         CALL CNVRT(%VAL(HDR2ADR),HDR2LEN,HDR2ADR)
C        TYPE *,HEAD
         CALL IDATE(IMONTH,IDAY,IYEAR)
         ENCODE(3,200,MONTH)IMONTH
         ENCODE(3,200,DAY  )IDAY
200      FORMAT(I3)
         ENCODE(2,100,YEAR )IYEAR
         HEAD(3)='OLDFAAD '
         HEAD(1)='USAMICOM'
         HEAD(2)=YEAR//MONTH//DAY
100      FORMAT(I2)
         ENCODE(8,200,WD)IWD
         HEAD(11)(6:8)=WD(1:3)
         ENCODE(8,200,LEN)ILEN
         HEAD(12)(6:8)=LEN(1:3)
C        TYPE *,HEAD
         IBRACKET=INDEX(FNAM,']')
         IPERIOD=INDEX(FNAM,'.')
         TNAME=FNAM(IPERIOD-9:IPERIOD-1)
         IBRACKET=IBRACKET
         IF(IPERIOD-10.LT.IBRACKET)THEN
         IZERO=ABS(IPERIOD-10)
         TNAME(1:IZERO)=' '
         ENDIF
         ILAST4=TNAME(6:9)
         IFIRST5=TNAME(1:5)

         HDR2ADR(1:8)=  'FN*X0000'
         HDR2ADR(11:18)='0000.IMG'
         HDR2ADR(4:8)=IFIRST5
         HDR2ADR(11:14)=ILAST4
         HDR2ADR(51:58)='SLREDALA'
         HDR2ADR(41:48)='LT000000'
         HDR2ADR(31:38)='RT000000'
         HDR2ADR(21:28)='DT000000'
C        HDR2ADR(351:358)=MILISECONDS
         CALL TIME(TIMEA)
         HDR2ADR(43:44)=TIMEA(1:2)
         HDR2ADR(45:46)=TIMEA(4:5)
         HDR2ADR(47:48)=TIMEA(7:8)
C        HDR2ADR(33:38)=HDR2ADR(43:48)
C        HDR2ADR(23:28)=HDR2ADR(43:48)
         HDR2ADR(23:28)=HEAD(2)(1:2)//HEAD(2)(4:5)//HEAD(2)(7:8)
```

[AVA.MAXDISK]MDSKTOFIL

```
        CALL UNCNVRT(%VAL(HDR2ADR),HDR2LEN,HDR2ADR)
C       TYPE*,'HDR2',HDR2ADR(1:HDR2LEN)
C       TYPE*,' WRITING ',FNAM(1:40)
        IHD2=HDR2LEN    !AUTOWRTSB ROUTINE NEEDS THIS DEFINED THROUGH AUTOIMG.CMN
        ISTATUS=AUTOWRTSB(1,1,ILEN,IWD,IMAGE,%VAL(HDR2ADR))
        IF(.NOT.ISTATUS)TYPE *,'ERROR IN AUTOWRTSB IMAGE TO DISK'
        RETURN
        END
```

APPENDIX I

[AVA.MAXDISK]MDSKTFIL2

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM READS THE DISK$IMAGES:[AVA]IMAGES.DAT FILE AND GENERATES
C       NATO FORMATTED DISK FILES FOR AS MANY FILES AS SPECIFIED BY THE USER.
C
C       IMAGES.DAT IS THE COMPACT IMAGE FORMAT.
C
C       THIS PROGRAM IS DIFFERENT FROM MDSKTOFIL IN THAT IT NEEDS
C       DISK$AVA:[AVA]IRIGS.DAT IN CORRECTED FORM. THE IRIGS IN IRIGS.DAT
C       ARE PUT IN THE RTXXYYZZ SECTION IN HEADER TWO IN THE CREATED FILE.
C       THE JULIAN DAY IS NOT CURRENTLY USED.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLS1,MITLS2
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*4 LIB$FREEVM,LIB$GETVM,SYS$DASSGN
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        BYTE BINPUT(32768)
        INTEGER AVACSR,AVAACR,OTS$CVTLTI
        INTEGER*2 ISETUP2(2),ISETUP3(2)
C        CHARACTER *8Ø MSGBUF,NAMNUM*4,DISKSPEC*14
        CHARACTER *8Ø MSGBUF,NAMNUM*4,DISKSPEC*22
        CHARACTER*6Ø TITLE,FNAME,FNAM2
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        COMMON/PRACHAN2/IDISK2
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
```

[AVA.MAXDISK]MDSKTFIL2

```
         1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
         2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
         DATA ISETUP2/'64777'O,'44ØØØ'O/
         DATA ISETUP3/'64776'O,'44ØØØ'O/
         DATA IFIRST/1/
         I=SYS$ASSIGN('TT',IVTC,,)
         IF(.NOT.I)TYPE *,'ERROR IN TT CHANNELL ASSIGN'
         I = SYS$ASSIGN('GRAØ',CHAN,,)
         IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
         ISTATUS=SYS$ASSIGN('AVAØ',ITCHAN,,)
         IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
         AVACSR=Ø
         AVAACR='415'O
         K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
         1BUF(1),%VAL(28),,,,)
         OPEN(UNIT=22,NAME='DISK$AVA:[AVA]IRIGS.DAT',STATUS='OLD',READONLY)
         TITLE=' READ DISK AND WRITE TO GRINNELL TIME'
         NIMAGES=1
         INSZ=NIMAGES*513
         FNAME='DISK$AVA:[AVA]IMAGES.DAT'
         OPEN(UNIT=3Ø,NAME=FNAME,TYPE='UNKNOWN',
         1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS2,
         2RECORDTYPE='FIXED',RECORDSIZE=4Ø96)
         IEVFO=4
         Y=6
         X=Ø
         TYPE *,'ENTER STARTING IMAGE NUMBER DESIRED'
         ACCEPT*,IBLOCK
         IF(IBLOCK.EQ.1)THEN
         IBLOCK=1
         ELSE
         IBLOCK=1+(48Ø*(IBLOCK-1))
         ENDIF
         ICOUNT=Ø
C        CALL TIMRB
         TYPE *,'ENTER NUMBER IMAGES TO STORE'
         ACCEPT*,IFIELDS
         IFIELDS=IFIELDS*8        !CONVERT IMAGES TO TRANSFERS
1        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
         NBYTES=24576
         ELSE
         NBYTES=32768
         ENDIF
         ISTATUS=SYS$QIOW(%VAL(IEVFO),%VAL(IDISK2),%VAL(%LOC(IO$READVBLK)),
         1IOSB,,,
         1BINPUT(1),%VAL(NBYTES),%VAL(IBLOCK),,,)
         IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
         IBLOCK=IBLOCK+48
         NUMB=24576
         ELSE
         NUMB=32768
         IBLOCK=IBLOCK+64
         ENDIF
C        ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C        1IOSB,,,
[AVA.MAXDISK]MDSKTFIL2

C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        IF(ICOUNT.EQ.3.OR.ICOUNT.EQ.7)THEN
        IGBYTES=24624
        ELSE
        IGBYTES=32832
        ENDIF

        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(IGBYTES),,,,)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(IGBYTES+1),%VAL(IGBYTES),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        IF(IFIRST)THEN
C       TYPE *,'ENTER DISK NAME AND DIRECTORY. (DISK$IMAGES:[WILLIAMS])'
C       READ(5,6)DISKSPEC
        DISKSPEC='DISK$AVA2:[AVA]'
C       DISKSPEC='DISK$AVA:[AVA]'
        TYPE *,'ENTER FIRST IMAGE FILE NAME. (WØØØ1ØØØØ.IMG)'
        READ(5,6)FNAM2
6       FORMAT(A)
C       FNAM='SIMSØØØØ1.IMG'
        NAMNUM=FNAM2(2:5)
        FNAM=DISKSPEC//FNAM2
        DECODE(4,1Ø1,NAMNUM)INAMNUM
1Ø1     FORMAT(I4)
        NCOL=512
        NROW=48Ø
        ILEN=NROW
        IWD=NCOL
C       IMGMAPC(3)=ILEN !LENGTH OF IMAGE
C       IMGMAPC(4)=IWD  !WIDTH  OF IMAGE
        IFIRST=Ø
        I=IWD
        NBYT=(I+1)*ILEN*2
        I=LIB$GETVM(NBYT,IMGADR)
        IF(.NOT.I)TYPE *,' ERROR IN VIRTUAL MEMORY ASSIGNMENT  1'
        I = LIB$GETVM(1ØØØØ,HDR2ADR)
        IF(.NOT. I) CALL ERRSTOP(I,'ERROR GETTING HDR2 VM','AVATODSK')
        ENDIF
        HEAD(8)='        1'       !ONE CHARACTER PER CHANNEL
        HDR2LEN=576
        CURRENTNUMFL=Ø
        CALL ADDHDR2(%VAL(HDR2ADR))
        CALL IMGTODISK(BINPUT,NUMB,%VAL(IMGADR),ICOUNT)
```

[AVA.MAXDISK]MDSKTFIL2

```
        IF(ICOUNT.EQ.7)THEN
C       FNAM='SIMSØØØØ1.IMG'
        INAMNUM=INAMNUM+1
        ISTATUS=OTS$CVTLTI(INAMNUM,NAMNUM,%VAL(4),%VAL(4),)
        IF(.NOT.ISTATUS)TYPE *,'CONVERSION ERROR IN FILE NAME'
        FNAM2(2:5)=NAMNUM
        FNAM=DISKSPEC//FNAM2
        ENDIF
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        ICOUNT=Ø
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP2(1),%VAL(4),,,,)
C       CALL TIMRE
C       CALL HEADER(TITLE)
C       STOP 'IMAGE READ IN AND DISPLAYED ON GRINNELL'
        Y=6
        ENDIF
        IFIELDS=IFIELDS-1
        IF(IFIELDS.EQ.Ø)STOP 'ALL IMAGES WRITTEN TO DISK'
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
```

```
[AVA.MAXDISK]MDSKTFIL2

C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END

        SUBROUTINE IMGTODISK(BINPUT,NUMB,IMAGE,ICOUNT)
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]DSP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]GRMAP.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
        INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
        INTEGER*2 IMAGE(NCOL+1,NROW)
        INTEGER*4 IMGADR,SYS$ASSIGN,IMGADR2,SYS$GETMSG
        BYTE BINPUT(1)
        INDEX=1
        IF(ICOUNT.LE.3)THEN
        ISTART=1+(ICOUNT*128)
        IF(ICOUNT.LE.2)IEND=ISTART+126
        IF(ICOUNT.EQ.3)IEND=ISTART+94
        DO I=ISTART,IEND,2
        DO J=1,NCOL
        INPUT=BINPUT(INDEX)
        IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C       IMAGE(J,I)=BINPUT(INDEX)
        INDEX=INDEX+1
        ENDDO
        ENDDO
        ELSE
        ISTART=2+((ICOUNT-4)*128)
        IF(ICOUNT.LE.6)IEND=ISTART+126
        IF(ICOUNT.EQ.7)IEND=ISTART+94
        DO I=ISTART,IEND,2
        DO J=1,NCOL
        INPUT=BINPUT(INDEX)
        IMAGE(J,I)=IAND('377'O,NOT(INPUT))
C       IMAGE(J,I)=BINPUT(INDEX)
        INDEX=INDEX+1
        ENDDO
        ENDDO
        ENDIF
        ILEN=NROW
        IWD=NCOL
C       TYPE*,'IWD AND ILEN BEFORE TODISK=',IWD,ILEN
        IF(ICOUNT.EQ.7) CALL TODISK(IMAGE,IWD,ILEN)
```

[AVA.MAXDISK]MDSKTFIL2

```
C        WRITE(6,1)ICOUNT
1        FORMAT(1X,I1Ø)
         RETURN
         END
         SUBROUTINE TODISK(IMAGE,IWD,ILEN)
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN/NOLIST'
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGNAME.CMN/NOLIST'
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]SUBCOM.CMN/NOLIST'
         INCLUDE 'DISK$USERDISK:[SUBIMAGE]AUTOIMG.CMN/NOLIST'
         INTEGER*2 IMAGE(NCOL+1,NROW),HDR2LEN
         INTEGER*4 AUTOWRTSB
         CHARACTER*1ØØØØ HDR2ADR
         CHARACTER*3 MONTH,DAY,YEAR*2,WD*8,LEN*8,TIMEA*8
         CHARACTER*5 IFIRST5,ILAST4*4,TNAME*9
         CHARACTER*2 HOUR,MINUTES,SECONDS,MSECONDS*3,HMS*12
         HDR2LEN=HDR2LEN
C        TYPE *,'HDR2LEN',HDR2LEN
C        TYPE *,'HDR2LEN ',HDR2LEN
         CALL CNVRT(%VAL(HDR2ADR),HDR2LEN,HDR2ADR)
C        TYPE *,HEAD
         CALL IDATE(IMONTH,IDAY,IYEAR)
         ENCODE(3,2ØØ,MONTH)IMONTH
         ENCODE(3,2ØØ,DAY   )IDAY
2ØØ      FORMAT(I3)
         ENCODE(2,1ØØ,YEAR )IYEAR
         HEAD(3)='OLDFAAD '
         HEAD(1)='USAMICOM'
         HEAD(2)=YEAR//MONTH//DAY
1ØØ      FORMAT(I2)
         ENCODE(8,2ØØ,WD)IWD
         HEAD(11)(6:8)=WD(1:3)
         ENCODE(8,2ØØ,LEN)ILEN
         HEAD(12)(6:8)=LEN(1:3)
C        TYPE *,HEAD
         IBRACKET=INDEX(FNAM,']')
         IPERIOD=INDEX(FNAM,'.')
         TNAME=FNAM(IPERIOD-9:IPERIOD-1)
         IBRACKET=IBRACKET
         IF(IPERIOD-1Ø.LT.IBRACKET)THEN
         IZERO=ABS(IPERIOD-1Ø)
         TNAME(1:IZERO)=' '
         ENDIF
         ILAST4=TNAME(6:9)
         IFIRST5=TNAME(1:5)

         HDR2ADR(1:8)=  'FN*XØØØØ'
         HDR2ADR(11:18)='ØØØØ.IMG'
         HDR2ADR(4:8)=IFIRST5
         HDR2ADR(11:14)=ILAST4
         HDR2ADR(51:58)='SLREDALA'
         HDR2ADR(41:48)='LTØØØØØØ'
         HDR2ADR(31:38)='RTØØØØØØ'
         HDR2ADR(21:28)='DTØØØØØØ'
         CALL TIME(TIMEA)
```

[AVA.MAXDISK]MDSKTFIL2

```
        HDR2ADR(43:44)=TIMEA(1:2)
        HDR2ADR(45:46)=TIMEA(4:5)
        HDR2ADR(47:48)=TIMEA(7:8)
C       HDR2ADR(33:38)=HDR2ADR(43:48)
C       HDR2ADR(23:28)=HDR2ADR(43:48)
        HDR2ADR(23:28)=HEAD(2)(1:2)//HEAD(2)(4:5)//HEAD(2)(7:8)
        READ(22,22)HOUR,MINUTES,SECONDS,MSECONDS
22      FORMAT(5X,A2,1X,A2,1X,A2,1X,A3)
        HDR2ADR(33:34)=HOUR
        HDR2ADR(35:36)=MINUTES
        HDR2ADR(37:38)=SECONDS
        HDR2ADR(279:281)=MSECONDS

        CALL UNCNVRT(%VAL(HDR2ADR),HDR2LEN,HDR2ADR)
C       TYPE*,'HDR2',HDR2ADR(1:HDR2LEN)
C       TYPE*,' WRITING ',FNAM(1:40)
        IHD2=HDR2LEN     !AUTOWRTSB ROUTINE NEEDS THIS DEFINED THROUGH AUTOIMG.CMN
        ISTATUS=AUTOWRTSB(1,1,ILEN,IWD,IMAGE,%VAL(HDR2ADR))
        IF(.NOT.ISTATUS)TYPE *,'ERROR IN AUTOWRTSB IMAGE TO DISK'
        RETURN
        END
```

APPENDIX J

[AVA]AVAFIELDS

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM ALLOWS THE USER TO SELECT WHICH FIELD IN THE AVA MEMORY
C       TO DISPLAY ON THE GRINNELL.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        BYTE BINPUT(32768)
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *80 MSGBUF
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24071'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP3/'64777'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=0
        AVAACR='415'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
12      TYPE *,'Enter the  AVA field number to be displayed on the GRINNELL.'
        TYPE *,'0,1,2, OR 3.'
        ACCEPT *,IFIELD
        IF(IFIELD.EQ.0)Y=6
        IF(IFIELD.EQ.1)Y='206'O
        IF(IFIELD.EQ.2)Y='406'O
```

[AVA]AVAFIELDS

```
        IF(IFIELD.EQ.3)Y='6Ø6'O
        IF(IFIELD.LT.Ø.OR.IFIELD.GT.3)GO TO 12
        X=Ø
        ICOUNT=Ø
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

        NUMB=32768
        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(65534),,,,)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP3(1),%VAL(4),,,,)
        GO TO 12
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
```

[AVA]AVAFIELDS

```
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

[AVA]AVAGROUP8

```
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,6Ø),X,Y
        BYTE BOUT(6156Ø),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(1536Ø)
        BYTE BINPUT(3Ø72Ø)
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
        1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
        2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
        DATA ISETUP2/'64777'O,'44ØØØ'O/
        DATA ISETUP3/'64776'O,'44ØØØ'O/
        I = SYS$ASSIGN('GRAØ',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=Ø
        AVAACR='415'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        Y=6
        X=Ø
        ICOUNT=Ø
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(3Ø72Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
```

```
54      FORMAT(1X,16(1X,O3))

        NUMB=30720
        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(61560),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+30
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        NUMB=8192
        CALL BUFFCNVT(NUMB,BINPUT,OUT)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(16384),,,,)
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP3(1),%VAL(4),,,,)
        Y='206'O
        X=0
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        ICOUNT=0
        ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        NUMB=8192
        CALL BUFFCNVT(NUMB,BINPUT,OUT)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(16384),,,,)
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP2(1),%VAL(4),,,,)
        Y=6
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
```

```
[AVA]AVAGROUP8

        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX L

[AVA]AVAGROUP9

```
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        BYTE BINPUT(32768)
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *80 MSGBUF
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64776'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=0
        AVAACR='415'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        Y=6
        X=0
        ICOUNT=0
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.0)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
```

```
[AVA]AVAGROUP9

54      FORMAT(1X,16(1X,O3))

        NUMB=32768
        CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(1),%VAL(65534),,,,)
        ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        ICOUNT=Ø
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
        K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1ISETUP2(1),%VAL(4),,,,)
        Y=6
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
```

[AVA]AVAGROUP9

```
C        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C        1ISETUP3,%VAL(4),,,,)
          END
         SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
         BYTE BINPUT(1),BYTE(2)
         INTEGER*2 OUT(513,1),BYTES,SLU
         EQUIVALENCE(BYTES,BYTE)
         DATA SLU/'34Ø11'O/
         I=Ø
         IOLINE=1
         DO 1ØØ IX=1,NUMB
         I=I+1
         IF(I.EQ.512)THEN
         BYTE(1)=BINPUT(IX)
         OUT(I,IOLINE)=BYTES
C        WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
         OUT(I+1,IOLINE)=SLU
C        WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
         I=Ø
         IOLINE=IOLINE+1
         GO TO 1ØØ
         ENDIF
         BYTE(1)=BINPUT(IX)
         OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C        WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34       FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ      CONTINUE
         RETURN
         END
```

APPENDIX M

[AVA]AVAFWRITE

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM ALLOWS THE USER TO WRITE A SPECIFIED BYTE TO A FIELD
C       IN THE AVA VIDEO MEMORY SELECTED BY HIM.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        INTEGER IERROR(512)
        BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
        INTEGER*2 IDATA
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *80 MSGBUF,TITLE
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64776'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=0
        AVAACR='435'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        ITESTN=1
        IERRORC=0
        DATAINA(1)='125'O
```

[AVA]AVAFWRITE

```
        DATAINA(2)=Ø
        DATAINA(3)='252'O
        DATAINA(4)='377'O
        DATAIN='125'O
        DATAIN=DATAINA(ITESTN)
57      TYPE *,'ENTER DATA TO BE WRITTEN INTO THE FIELD IN OCTAL. (I.E. 377)'
        READ(5,56)DATAIN
56      FORMAT(O3)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
        TYPE *,'ENTER THE FIELD TO WRITE THE DATA INTO. (Ø,1,2, OR 3)'
        READ(5,56)IFIELD
        IF(IFIELD.EQ.Ø)Y=6
        IF(IFIELD.EQ.1)Y='2Ø6'O
        IF(IFIELD.EQ.2)Y='4Ø6'O
        IF(IFIELD.EQ.3)Y='6Ø6'O
        IF(IFIELD.LT.Ø.OR.IFIELD.GT.3)GO TO 57
C177    Y=6
        X=Ø
        ICOUNT=Ø
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        GO TO 57
        ENDIF
        GO TO 1
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
```

[AVA]AVAFWRITE

```
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX N

[AVA]RAMPMAX

```
      EXTERNAL IO$WRITEVBLK,IO$READVBLK
      INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
      INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
      INTEGER SYS$GETMSG,MSGLEN,ISTATUS
      INTEGER*2 OUT(513,64),X,Y
      BYTE BOUT(65664),BYTE(2)
      INTEGER*2 BYTES
      INTEGER*2 OUTPUT,INIT(4)
      INTEGER*2 INPUT(16384)
      BYTE BINPUT(32768),BDATA(2)
      INTEGER*2 IDATA
      INTEGER AVACSR,AVAACR
      INTEGER*2 ISETUP2(2),ISETUP3(2)
      CHARACTER *80 MSGBUF,TITLE
      EQUIVALENCE(BUF(1),ISETUP(1))
      EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
      EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
      DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
      1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
      2 '50001'O,'70776'O,'54000'O/
      DATA ISETUP2/'64777'O,'44000'O/
      DATA ISETUP3/'64776'O,'44000'O/
      I = SYS$ASSIGN('GRA0',CHAN,,)
      IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
      ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
      IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
      AVACSR=0
      AVAACR='435'O
      K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
      1BUF(1),%VAL(28),,,,)
      Y=6
      X=0
      ICOUNT=0
      DO I=1,32768
      IDATA=IAND((I-1),'377'O)
      BINPUT(I)=BDATA(1)
      ENDDO
      TITLE='FOUR FIELD WRITE TO AVA'
      CALL TIMRB
1     ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP2(1),%VAL(4),,,,)
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        ENDIF
        IF(ICOUNT.EQ.24)THEN
        ICOUNT=Ø
        CALL  TIMRE
        CALL HEADER(TITLE)
```

[AVA]RAMPMAX

```
        Y=6
        CALL TIMRB
        ENDIF

        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX O

[AVA]RAMPMAX2

```
EXTERNAL IOSWRITEVBLK,IOSREADVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
INTEGER SYS$GETMSG,MSGLEN,ISTATUS
PARAMETER NLINES=64
INTEGER*2 OUT(513,NLINES),X,Y
BYTE BOUT(1026*NLINES),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(256*NLINES)
BYTE BINPUT(512*NLINES),BDATA(2)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYS$ASSIGN('GRA0',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IOSWRITEVBLK)),IOSB,,,
1BUF(1),%VAL(28),,,,)
Y=6
X=0
ICOUNT=0
TITLE='AVA READ FRAME TIME'
DO I=1,512*NLINES
IDATA=IAND((I-1),'377'O)
BINPUT(I)=BDATA(1)
ENDDO
CALL TIMRB
```

```
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(512*NLINES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=512*NLINES
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(36866),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+(NLINES/2)
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        ICOUNT=Ø
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP2(1),%VAL(4),,,,)
        CALL TIMRE
        CALL HEADER(TITLE)
        CALL TIMRB
        Y=6
        ENDIF
        GO TO 1
57      CONTINUE
```

- - --

[AVA]RAMPMAX2

```
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX P

[AVA.MAXDISK]MTODSKT

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM IS USED TO FIND OUT WHAT DISK WRITE TIMING IS REQUIRED
C       TO ALLOW EVERY FIELD TO BE WRITTEN TO DISK.
C
C       NO I/O TO DISK OCCURS IN THIS PROGRAM! THE DISK I/O TIME IS SIMULATED
C       WITH A DELAY ROUTINE.
C
C       THE AVA READS STILL HAPPEN SO THE HBR-3ØØØ MUST BE ON AND RUNNING
C       AT 3 3/4 OR 1 7/8.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK,MITLSI
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        CHARACTER*16 TIME,TIME2
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 X,Y,YA(4),SYS$DASSGN
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(65536)
C       BYTE BINPUT(32768)
        BYTE BINPUT(131Ø72)
        INTEGER AVACSR,AVAACR,SYS$LKWSET,INLOCK(2),IOLOCK(2)
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF
        CHARACTER*6Ø TITLE,FNAME*6Ø
        CHARACTER*6Ø NAME
        INTEGER*2 BUFFERL,DEVCODE
        INTEGER SYS$GETDVI,DVI$FREEBLOCKS
        INTEGER IFREE
        INTEGER BUFFERA,ZERO
        COMMON/PRACHAN/IDISK
        COMMON/ITEMLIST/BUFFERL,DEVCODE,BUFFERA,ZERO
        COMMON/AVACHAN/ITCHAN
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT)
        DATA TIME /'ØØØØ ØØ:ØØ:ØØ.Ø2'/
        DATA TIME2/'ØØØØ ØØ:ØØ:ØØ.Ø1'/
        DATA YA/6,'2Ø6'O,'4Ø6'O,'6Ø6'O/
```

[AVA.MAXDISK]MTODSKT

```
        DATA DVI$FREEBLOCKS/'0000002A'X/,ZERO/0/
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64776'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        TITLE=' READ AVA BUFFER AND WRITE TO DISK TIME'
C       NAME='DISK$AVA:'
C       BUFFERL=4
C       DEVCODE=DVI$FREEBLOCKS
C       BUFFERA=%LOC(IFREE)
C       RETURNLA=%LOC(RETURNL)
C       ISTATUS=SYS$GETDVI(%VAL(3),,NAME,BUFFERL,,,,)
C
C       IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN GETDVI'
C
C       ISTATUS=SYS$WAITFR(%VAL(3))
C       TYPE *,'BLOCKS FREE FOR IMAGE STORAGE=',IFREE
C       MAXIMAGES=IFREE/513
C
C       TYPE *,'MAXIMUM NUMBER IMAGES THAT CAN BE STORED=',MAXIMAGES
C
C
        MAXIMAGES=50     !THIS IS FOR DEGUG ONLY
C
C       NIMAGES=MAXIMAGES
C       INSZ=NIMAGES*513
C       FNAME='DISK$AVA:[AVA]IMAGES.DAT'
C       OPEN(UNIT=30,NAME=FNAME,TYPE='UNKNOWN',
C       1FORM='UNFORMATTED',INITIALSIZE=INSZ,USEROPEN=MITLS1,
C       2RECORDTYPE='FIXED',RECORDSIZE=4096)
        ISTATUS=SYS$ASSIGN('AVA0',AVACHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        INLOCK(1)=%LOC(BINPUT(1))
        INLOCK(2)=%LOC(BINPUT(131072))
        K=SYS$LKWSET(INLOCK,IOLOCK,)
        TYPE *,' INLOCK(1)= ',INLOCK(1),' INLOCK(2)= ',INLOCK(2)
        TYPE *,' IOLOCK(1)= ',IOLOCK(1),' IOLOCK(2)= ',IOLOCK(2)
        IF(.NOT.K)TYPE *,' UNABLE TO LOCK BUF'
        AVACSR=0
        AVAACR='415'O
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BUF(1),%VAL(28),,,,)
        IEVFO=4
        IMAGEN=1
        IBLOCK=1
        CALL TIMRB
        X=0
        CALL FIELD(IFIELD,AVACSR)
        ICURR=IFIELD
C       TYPE *,'ICURR=',ICURR
```

```
[AVA.MAXDISK]MTODSKT

11      CALL FIELD(IFIELD,AVACSR)
        IF(IFIELD.EQ.ICURR)GO TO 11
        ISTOREFIELD=ICURR       !CURRENT FIELD TO PUT ON DISK
        ICURR=IFIELD            !CURRENT FIELD BE LOADED INTO THE AVA
C       TYPE *,'ICURR=',ICURR,'ISTOREFIELD=',ISTOREFIELD
C       GO TO 11
        Y=YA(ISTOREFIELD+1)
C       ICOUNT=Ø
1       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
        CALL DELAY(TIME)
C       ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
C       1IOSB,,,
C       1BINPUT(1),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
2       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(32679),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        CALL DELAY(TIME)
C       ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
C       1IOSB,,,
C       1BINPUT(32679),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
3       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(65537),%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        CALL DELAY(TIME)
C       ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
C       1IOSB,,,
C       1BINPUT(65537),%VAL(32768),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+64
        Y=Y+32
4       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1BINPUT(983Ø5),%VAL(24576),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        CALL DELAY(TIME2)
C       ISTATUS=SYS$QIO(%VAL(IEVFO),%VAL(IDISK),%VAL(%LOC(IO$WRITEVBLK)),
C       1IOSB,,,
C       1BINPUT(983Ø5),%VAL(24576),%VAL(IBLOCK),,,)
        IBLOCK=IBLOCK+48
C       IF(AVACSR.EQ.Ø)AVACSR=1

C       WRITE (4,54)BINPUT
54      FORMAT(1X 16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
```

[AVA.MAXDISK]MTODSKT .

```
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       ICOUNT=ICOUNT+1
C       IF(ICOUNT.NE.4)GO TO 1
        CALL FIELD(IFIELD,AVACSR)
                IF(IFIELD.EQ.ICURR)THEN
        IF(IMAGEN.EQ.MAXIMAGES)THEN
        ISTATUS=SYS$WAITFR(%VAL(1))
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        TYPE *,' I/O COMPLETE..............................'
        CALL TIMRE
        CALL HEADER(TITLE)
        ISTATUS=SYS$DASSGN(%VAL(IDISK))
        CLOSE(UNIT=3Ø)
        TYPE *,IMAGEN,' FIELDS WRITTEN TO DISK'
        STOP 'ALL IMAGES WRITTEN TO DISK'
        ENDIF
        IMAGEN=IMAGEN+1
                GO TO 11
                ELSE
                TYPE*,'FATAL ERROR....****....I/O TO SLOW'
                TYPE *,'BLOCK NUMBER=',IBLOCK
                TYPE *,'ICURR=',ICURR,' IFIELD=',IFIELD
                TYPE *,IMAGEN,' FIELDS WRITTEN TO DISK'
                CALL TIMRE
                CALL HEADER(TITLE)
                ISTATUS=SYS$DASSGN(%VAL(IDISK))
                CLOSE(UNIT=3Ø)
                STOP
                ENDIF
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE DELAY(TIME)
        DOUBLE PRECISION QUAD
        INTEGER SYS$BINTIM,SYS$SETIMR,SYS$WAITFR
        CHARACTER*16 TIME
C       RETURN
C       TIME='ØØØØ ØØ:ØØ:ØØ.5Ø'
        ISTATUS=SYS$BINTIM(%DESCR(TIME),QUAD)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        ISTATUS=SYS$SETIMR(%VAL(6),QUAD,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        ISTATUS=SYS$WAITFR(%VAL(6))
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        RETURN
        END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34011'O/
        I=0
        IOLINE=1
        DO 100 IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=0
        IOLINE=IOLINE+1
        GO TO 100
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
100     CONTINUE
        RETURN
        END

        SUBROUTINE FIELD(IFIELD,AVACSR)
        INTEGER AVACSR
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
        INTEGER*2 INPUT,OUTPUT,INIT(4)
        CHARACTER *80 MSGBUF
        COMMON/AVACHAN/ITCHAN
        DATA IFIRST/1/
        ISAVE=AVACSR
        IF(IFIRST)THEN
        AVACSR='40000'O  !SET MEMORY WINDOW ENABLE AND INITIALIZE AVA
        IFIRST=0
        ELSE
        AVACSR='40001'O
        ENDIF
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT,%VAL(2),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(IAVAACR))
C       IF(AVACSR.EQ.'40000'O)AVACSR='40001'O
C       IF(ISTATUS)     GO TO 501
C       TYPE *,' ERROR IN QIOW CALL'
C       ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
```

```
[AVA.MAXDISK]MTODSKT

C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C       MSGBUF=' '
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'I/O STATUS:',MSGBUF
5Ø1     AVACSR=ISAVE
        IFIELD=IAND(OUTPUT,3)
        RETURN
        END
```

APPENDIX Q

[AVA]FAVAMEMT

```
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
INTEGER SYS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYS$ASSIGN('GRA0',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(28),,,,)
ITESTN=1
IERRORC=0
DATAINA(1)='125'O
DATAINA(2)=0
DATAINA(3)='252'O
DATAINA(4)='377'O
DATAIN='125'O
DATAIN=DATAINA(ITESTN)
DO I=1,32768
```

```
        BINPUT(I)=DATAIN
        ENDDO
177     Y=6
        X=Ø
        ICOUNT=Ø
        TITLE='FOUR FIELD WRITE TO AVA'
C       CALL TIMRB
I       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP2(1),%VAL(4),,,,)
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        ENDIF
        IF(ICOUNT.EQ.24)THEN
CCCCCCCCCCC   READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME  CCCCCC
C
        ICOUNT=Ø
        Y=6
        X=Ø
51      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
        DO IADD=1,32768
        IF(BINPUT(IADD).NE.DATAIN)THEN
        IIADD=IAND(IADD,'777'O)
C       WRITE(6,12)DATAIN,BINPUT(IADD),
C       1IAND(IADD       ,'777'O)
        IERROR(IIADD)=IERROR(IIADD)+1
        IERRORC=IERRORC+1
        ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',O3,2X,'OUTPUT= ',O3,
        15X,'COLUMN= ',I5)
        ENDDO
C       TYPE *,' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT


        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        WRITE(6,56)IERRORC,DATAIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        WRITE(6,157)IERRORC,DATAIN
157     FORMAT(1X,' ERROR COUNT IN FIELD 2=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        WRITE(6,58)IERRORC,DATAIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        ENDIF
        IF(ICOUNT.EQ.24)THEN
        WRITE(6,59)IERRORC,DATAIN
59      FORMAT(1X,' ERROR COUNT IN FIELD 4=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        DO IADD=1,512
        IF(IERROR(IADD).NE.Ø)WRITE(6,233)IADD,
```

[AVA]FAVAMEMT

```
        1IERROR(IADD),DATAIN
233     FORMAT(1X,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=Ø
        DO IADD=1,512
        IERROR(IADD)=Ø
        ENDDO
        ICOUNT=Ø
        Y=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
        GC TO 177
        ENDIF
        GO TO 51
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4);,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
```

[AVA]FAVAMEMT

```
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
100     CONTINUE
        RETURN
        END
```

APPENDIX R

[AVA]FAVAMEMT2

```
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
INTEGER SYS$GETMSG,MSGLEN,ISTATUS
INTEGER*2 OUT(513,64),X,Y
BYTE BOUT(65664),BYTE(2)
INTEGER*2 BYTES
INTEGER*2 OUTPUT,INIT(4)
INTEGER*2 INPUT(16384)
INTEGER IERROR(512)
BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
INTEGER*2 IDATA
INTEGER AVACSR,AVAACR
INTEGER*2 ISETUP2(2),ISETUP3(2)
CHARACTER *80 MSGBUF,TITLE
EQUIVALENCE(BUF(1),ISETUP(1))
EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
2 '50001'O,'70776'O,'54000'O/
DATA ISETUP2/'64777'O,'44000'O/
DATA ISETUP3/'64776'O,'44000'O/
I = SYS$ASSIGN('GRA0',CHAN,,)
IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
AVACSR=0
AVAACR='435'O
K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(28),,,,)
ITESTN=1
IERRORC=0
DATAINA(1)='125'O
DATAINA(2)=0
DATAINA(3)='252'O
DATAINA(4)='377'O
DATAIN='125'O
DATAIN=DATAINA(ITESTN)
DO I=1,32768
```

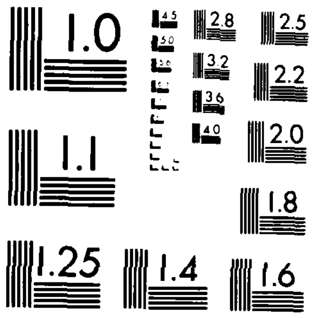
```
        BINPUT(I)=DATAIN
        ENDDO
177     Y=6
        X=Ø
        ICOUNT=Ø
        TITLE='FOUR FIELD WRITE TO AVA'
C       CALL TIMRB
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VA.'1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP2(1),%VAL(4),,,,)
        Y='4Ø6'O
```

AD-A146 249 ON LINE DIGITIZER SOFTWARE(U) ARMY MISSILE COMMAND REDSTONE ARSENAL AL ADVANCED SENSORS DIRECTORATE S R SIMS JUN 84 DRSMI/RE-84-17-TR SBI-AD-E950 550

2/2

UNCLASSIFIED F/G 9/2 NL

END
DATE
FILMED
10-84
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

```
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='606'O
        ENDIF
        IF(ICOUNT.EQ.24)THEN
CCCCCCCCCCC   READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME  CCCCCC
C
        ICOUNT=0
        Y=6
        X=0
51      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.0)AVACSR=1
        DO IADD=1,32768
        IF(BINPUT(IADD).NE.DATAIN)THEN
        IIADD=IAND(IADD,'777'O)
        WRITE(6,12)DATAIN,BINPUT(IADD),
        1IAND(IADD       ,'777'O)
        IERROR(IIADD)=IERROR(IIADD)+1
        IERRORC=IERRORC+1
        ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',O3,2X,'OUTPUT= ',O3,
        15X,'COLUMN= ',I5)
        ENDDO
C       TYPE *,' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT


        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        WRITE(6,56)IERRORC,DATAIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',I5,' DATA IN= ',O3)
        IERRORC=0
        Y='206'O
        X=0
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        WRITE(6,157)IERRORC,DATAIN
157     FORMAT(1X,' ERROR COUNT IN FIELD 2=',I5,' DATA IN= ',O3)
        IERRORC=0
        Y='406'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='606'O
        WRITE(6,58)IERRORC,DATAIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',I5,' DATA IN= ',O3)
        IERRORC=0
        ENDIF
        IF(ICOUNT.EQ.24)THEN
        WRITE(6,59)IERRORC,DATAIN
59      FORMAT(1X,' ERROR COUNT IN FIELD 4=',I5,' DATA IN= ',O3)
        IERRORC=0
        DO IADD=1,512
        IF(IERROR(IADD).NE.0)WRITE(6,233)IADD,
```

[AVA]FAVAMEMT2

```
        1IERROR(IADD),DATAIN
233     FORMAT(1X,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=Ø
        DO IADD=1,512
        IERROR(IADD)=Ø
        ENDDO
        ICOUNT=Ø
        Y=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
        ENDIF
        GO TO 51
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
```

[AVA]FAVAMEMT2

```
      OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C     WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34    FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ   CONTINUE
      RETURN
      END
```

APPENDIX S

[AVA]AVAMEMT

```
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER*2 BUF(2ØØ),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        INTEGER IERROR(512)
        BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
        BYTE BYTEDAT
        INTEGER*2 IDATA
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *8Ø MSGBUF,TITLE
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'12ØØ4Ø'O,'14ØØØ1'O,'121ØØØ'O,'1Ø7777'O,'17777'O,
        1 '24Ø61'O,'26ØØ2'O,'3ØØØØ'O,'44ØØØ'O,'64777'O,'12ØØØØ'O,
        2 '5ØØØ1'O,'7Ø776'O,'54ØØØ'O/
        DATA ISETUP2/'64777'O,'44ØØØ'O/
        DATA ISETUP3/'64776'O,'44ØØØ'O/
        I = SYS$ASSIGN('GRAØ',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=Ø
        AVAACR='435'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        ITESTN=1
        IERRORC=Ø
        TYPE *,' ENTER MEMORY DATA FOR TESTING'
        READ(5,78)BYTEDAT
78      FORMAT(O3)
        DATAINA(1)=BYTEDAT
        DATAINA(2)=BYTEDAT
        DATAINA(3)=BYTEDAT
```

[AVA]AVAMEMT

```
        DATAINA(4)=BYTEDAT
C       DATAIN='125'O
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
177     Y=6
        X=Ø
        ICOUNT=Ø
        TITLE='FOUR FIELD WRITE TO AVA'
C       CALL TIMRB
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP2(1),%VAL(4),,,,)
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        ENDIF
        IF(ICOUNT.EQ.24)THEN
CCCCCCCCCCC   READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME  CCCCCC
C
        ICOUNT=Ø
        Y=6
        X=Ø
51      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
        DO IADD=1,32768
        IF(BINPUT(IADD).NE.DATAIN)THEN
        IIADD=IAND(IADD,'777'O)
C       WRITE(6,12)DATAIN,BINPUT(IADD),
C       1IAND(IADD      ,'777'O)
        IERROR(IIADD)=IERROR(IIADD)+1
        IERRORC=IERRORC+1
        ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',O3,2X,'OUTPUT= ',O3,
        15X,'COLUMN= ',I5)
        ENDDO
C       TYPE *,' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT


        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        WRITE(6,56)IERRORC,DATAIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        WRITE(6,157)IERRORC,DATAIN
157     FORMAT(1X,' ERROR COUNT IN FIELD 2=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        WRITE(6,58)IERRORC,DATAIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        ENDIF
        IF(ICOUNT.EQ.24)THEN
        WRITE(6,59)IERRORC,DATAIN
```

```
59      FORMAT(1X,' ERROR COUNT IN FIELD 4=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        DO IADD=1,512
        IF(IERROR(IADD).NE.Ø)WRITE(6,233)IADD,
        1IERROR(IADD),DATAIN
233     FORMAT(1X,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=Ø
        DO IADD=1,512
        IERROR(IADD)=Ø
        ENDDO
        ICOUNT=Ø
        Y=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
        ENDIF
        GO TO 51
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
```

[AVA]AVAMENT

```
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX T

[AVA]AVAMEMT2

```
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        INTEGER IERROR(512)
        BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
        BYTE BYTEDAT
        INTEGER*2 IDATA
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *80 MSGBUF,TITLE
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64776'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=0
        AVAACR='435'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        ITESTN=1
        IERRORC=0
        TYPE *,' ENTER MEMORY DATA FOR TESTING'
        READ(5,78)BYTEDAT
78      FORMAT(O3)
        DATAINA(1)=BYTEDAT
        DATAINA(2)=BYTEDAT
        DATAINA(3)=BYTEDAT
```

[AVA]AVAMEMT2

```
        DATAINA(4`=BYTEDAT
C       DATAIN='125'O
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
177     Y=6
        X=Ø
        ICOUNT=Ø
        TITLE='FOUR FIELD WRITE TO AVA'
C       CALL TIMRB
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3(1),%VAL(4),,,,)
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
C       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
C       1IOSB,,,
C       1INPUT,%VAL(8192),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
C       NUMB=8192
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
```

[AVA]AVAMEMT2

```
C       1BOUT(1),%VAL(8192),,,,)
C       K = SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP2(1),%VAL(4),,,,)
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        ENDIF
        IF(ICOUNT.EQ.24)THEN
CCCCCCCCCCC   READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME  CCCCCC
C
        ICOUNT=Ø
        Y=6
        X=Ø
51      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
        DO IADD=1,32768
        IF(BINPUT(IADD).NE.DATAIN)THEN
        IIADD=IAND(IADD,'777'O)
        WRITE(6,12)DATAIN,BINPUT(IADD),
        1IAND(IADD       ,'777'O)
        IERROR(IIADD)=IERROR(IIADD)+1
        IERRORC=IERRORC+1
        ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',O3,2X,'OUTPUT= ',O3,
        15X,'COLUMN= ',I5)
        ENDDO
C       TYPE *,' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT


        Y=Y+32
        ICOUNT=ICOUNT+1
        IF(ICOUNT.EQ.4)THEN
        WRITE(6,56)IERRORC,DATAIN
56      FORMAT(1X,' ERROR COUNT IN FIELD 1=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        Y='2Ø6'O
        X=Ø
        ENDIF
        IF(ICOUNT.EQ.8)THEN
        WRITE(6,157)IERRORC,DATAIN
157     FORMAT(1X,' ERROR COUNT IN FIELD 2=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        Y='4Ø6'O
        ENDIF
        IF(ICOUNT.EQ.16)THEN
        Y='6Ø6'O
        WRITE(6,58)IERRORC,DATAIN
58      FORMAT(1X,' ERROR COUNT IN FIELD 3=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        ENDIF
        IF(ICOUNT.EQ.24)THEN
        WRITE(6,59)IERRORC,DATAIN
```

```
59      FORMAT(1X,' ERROR COUNT IN FIELD 4=',I5,' DATA IN= ',O3)
        IERRORC=Ø
        DO IADD=1,512
        IF(IERROR(IADD).NE.Ø)WRITE(6,233)IADD,
        1IERROR(IADD),DATAIN
233     FORMAT(1X,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=Ø
        DO IADD=1,512
        IERROR(IADD)=Ø
        ENDDO
        ICOUNT=Ø
        Y=6
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
        ENDIF
        GO TO 51
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
```

[AVA]AVAMEMT2

```
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX U

[AVA]AAVAMEMT

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM TESTS ALL OF THE AVA MEMORY (NOT JUST WHERE VIDEO IS STORED)
C       UP TO Y='777'O AND X='1777'O
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IOSWRITEVBLK,IOSREADVBLK
        INTEGER*2 BUF(200),ISETUP(14),SLU,IOSB(4)
        INTEGER    SYS$ASSIGN, SYS$QIOW, CHAN,SYS$QIO,SYS$WAITFR
        INTEGER SYS$GETMSG,MSGLEN,ISTATUS
        INTEGER*2 OUT(513,64),X,Y
        BYTE BOUT(65664),BYTE(2)
        INTEGER*2 BYTES
        INTEGER*2 OUTPUT,INIT(4)
        INTEGER*2 INPUT(16384)
        INTEGER IERROR(512)
        BYTE BINPUT(32768),BDATA(2),DATAIN,DATAINA(4)
        INTEGER*2 IDATA
        INTEGER AVACSR,AVAACR
        INTEGER*2 ISETUP2(2),ISETUP3(2)
        CHARACTER *80 MSGBUF,TITLE
        EQUIVALENCE(BUF(1),ISETUP(1))
        EQUIVALENCE(BINPUT,INPUT),(IDATA,BDATA)
        EQUIVALENCE(BOUT,OUT),(BYTE,BYTES)
        DATA ISETUP/'120040'O,'140001'O,'121000'O,'107777'O,'17777'O,
        1 '24061'O,'26002'O,'30000'O,'44000'O,'64777'O,'120000'O,
        2 '50001'O,'70776'O,'54000'O/
        DATA ISETUP2/'64777'O,'44000'O/
        DATA ISETUP3/'64776'O,'44000'O/
        I = SYS$ASSIGN('GRA0',CHAN,,)
        IF(.NOT. I)TYPE *,' ERROR IN GRINNELL CHANNEL ASSIGN'
        ISTATUS=SYS$ASSIGN('AVA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
        AVACSR=0
        AVAACR='435'O
        K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IOSWRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(28),,,,)
        ITESTN=1
        IERRORC=0
        DATAINA(1)='125'O
```

[AVA]AAVAMEMT

```
        DATAINA(2)=Ø
        DATAINA(3)='252'O
        DATAINA(4)='377'O
        DATAIN='125'O
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
177     Y=Ø
        X=Ø
        ICOUNT=Ø
        TITLE='FOUR FIELD WRITE TO AVA'
C       CALL TIMRB
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
C       1INPUT,%VAL(NBYTES),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57

C       WRITE (4,54)BINPUT
54      FORMAT(1X,16(1X,O3))

C       NUMB=32768
C       CALL BUFFCNVT(NUMB,BINPUT,OUT)
C       TYPE *,'NUMBER OF LINES TO OUTPUT=',IOLINE
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(1),%VAL(65534),,,,)
C       ISTATUS = SYS$QIO(%VAL(1),%VAL(CHAN),
C       1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1BOUT(65535),%VAL(13Ø),,,,)
C       IF(.NOT.ISTATUS.OR..NOT.IOSB(1))GO TO 57
        Y=Y+32
        IF(Y.GT.'777'O)THEN
CCCCCCCCCCC   READ AVA BACK NOW AND SEE IF THE DATA IS THE SAME  CCCCCC
C
        ICOUNT=Ø
        Y=Ø
        X=Ø
51      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1INPUT,%VAL(32768),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.Ø)AVACSR=1
        DO IADD=1,32768
        IF(BINPUT(IADD).NE.DATAIN)THEN
        IIADD=IAND(IADD,'777'O)
C       WRITE(6,12)DATAIN,BINPUT(IADD),
C       1IAND(IADD      ,'777'O)
        IERROR(IIADD)=IERROR(IIADD)+1
        IERRORC=IERRORC+1
        ENDIF
12      FORMAT(1X,'AVA MEMORY ERROR. INPUT= ',O3,2X,'OUTPUT= ',O3,
        15X,'COLUMN= ',I5)
        ENDDO
```

```
C       TYPE *,' ERROR COUNT AFTER FIELD ***=',IERRORC,ICOUNT


        Y=Y+32
        IF(Y.GT.'777'O)THEN
        WRITE(6,59)IERRORC,DATAIN
59      FORMAT(1X,' ERROR COUNT =',I5,' DATA IN= ',O3)
        IERRORC=Ø
        DO IADD=1,512
        IF(IERROR(IADD).NE.Ø)WRITE(6,233)IADD,
        1IERROR(IADD),DATAIN
233     FORMAT(1X,'COLUMN',I4,' ERRORS= ',I6,' DATA IN= ',O3)
        ENDDO
        IERRORC=Ø
        DO IADD=1,512
        IERROR(IADD)=Ø
        ENDDO
        ICOUNT=Ø
        Y=Ø
        ITESTN=ITESTN+1
        IF(ITESTN.GT.4)ITESTN=1
        DATAIN=DATAINA(ITESTN)
        DO I=1,32768
        BINPUT(I)=DATAIN
        ENDDO
        GO TO 177
        ENDIF
        GO TO 51
        ENDIF
        GO TO 1
57      CONTINUE
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,'  IOSB(1)=',IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        STOP
C11     FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       K = SYS$QIOW(%VAL(1),%VAL(CHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
C       1ISETUP3,%VAL(4),,,,)
         END
        SUBROUTINE BUFFCNVT(NUMB,BINPUT,OUT)
        BYTE BINPUT(1),BYTE(2)
        INTEGER*2 OUT(513,1),BYTES,SLU
        EQUIVALENCE(BYTES,BYTE)
        DATA SLU/'34Ø11'O/
        I=Ø
        IOLINE=1
        DO 1ØØ IX=1,NUMB
        I=I+1
        IF(I.EQ.512)THEN
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=BYTES
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
        OUT(I+1,IOLINE)=SLU
C       WRITE(6,34) I+1,IOLINE,OUT(I+1,IOLINE)
        I=Ø
        IOLINE=IOLINE+1
        GO TO 1ØØ
        ENDIF
        BYTE(1)=BINPUT(IX)
        OUT(I,IOLINE)=IAND(NOT(BYTES),'377'O)
C       WRITE(6,34) I,IOLINE,OUT(I,IOLINE)
34      FORMAT(1X,I3,1X,I3,2X,O6)
1ØØ     CONTINUE
        RETURN
        END
```

APPENDIX V

[AVA]AUX

```
      EXTERNAL IO$READVBLK,IO$WRITEVBLK
      BYTE IX(1ØØ),IY(1ØØ),N,IFLAG
      INTEGER*2 D(1ØØ),X,Y,IOSB(4)
      INTEGER*2 OUT(16),US,TS,UM,TM,UH21
      DIMENSION VOLTS(16)
      BYTE BD(2ØØ),TEMP,IIMAGE8(4)
      INTEGER*2 UH84,UH,TH,UD,TD,HD
      CHARACTER*8Ø MSGBUF
      INTEGER*2 TEMS,MS,HMS,TMS
      INTEGER AVAACR
      INTEGER AVACSR,SYS$ASSIGN,SYS$GETMSG,SYS$QIOW,SYS$QIO
      EQUIVALENCE(D,BD)
      ISTATUS=SYS$ASSIGN('AVAØ',IAVAC,,)
      IF(.NOT.ISTATUS)THEN
      TYPE *,' AVA CHANNEL ASSIGN ERROR'
      STOP
      ENDIF
      IFLAG=1
      AVAACR='415'O
      AVACSR=Ø
3     Y=1
      X=Ø
      ISTATUS=SYS$QIOW(%VAL(1),%VAL(IAVAC),%VAL(%LOC(IO$READVBLK)),
     1IOSB,,,
     1D,%VAL(36),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
              IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
              ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
              IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
              TYPE *,MSGBUF
      MSGBUF=' '
              ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
              IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
              TYPE *,MSGBUF
              STOP
      ENDIF
      IF(AVACSR.EQ.Ø)AVACSR=1
      DO I=3,35,2
      TEMP=BD(I)
      BD(I)=BD(I+1)
      BD(I+1)=TEMP
```

[AVA]AUX

```
        ENDDO
C       WRITE(6,111)(D(I),I=1,16)
111     FORMAT(1X,16(1X,Z4))
        IF(IFLAG)THEN
        IFLAG=Ø
        IIMAGEB(1)=27    !ESC
        IIMAGEB(2)=72    !H       CURSOR HOME
        IIMAGEB(3)=27    !ESC
        IIMAGEB(4)=74    !J       ERASE TO END OF SCREEN
        WRITE(6,77)IIMAGEB
        IIMAGEB(1)=27    !ESC
        IIMAGEB(2)=89    !Y
        IIMAGEB(4)=32    !COLUMN
        IIMAGEB(3)=37    !LINE
        WRITE(6,177)IIMAGEB
177     FORMAT(1H+,4A1,' CHANNEL           OCTAL               HEX',
        1'            VOLTS')
        ENDIF
        DO I=2,17
        ICHAN=IAND(NOT(ISHFT(D(I),-12))-1,'F'X)
        IS=IAND(ISHFT(D(I),-11),1)
        IF(IS.EQ.Ø)THEN
        II=FLOAT(D(I))+1.
        ELSE
        II=D(I)
        ENDIF
        OUT(ICHAN+1)=IAND(II,'7777'O)
        VOLTS(ICHAN+1)=FLOAT(IAND(II,'3777'O))/2Ø47.
        VOLTS(ICHAN+1)=VOLTS(ICHAN+1)*1Ø.
        IF(IS.NE.Ø)VOLTS(ICHAN+1)=-VOLTS(ICHAN+1)
        IF(VOLTS(ICHAN+1).GT.Ø.)VOLTS(ICHAN+1)=ABS(VOLTS(ICHAN+1)-1Ø.)
        ENDDO
        DO I=1,16
        IIMAGEB(3)=I+39 !LINE
C       WRITE(6,77)IIMAGEB,(OUT(I),I=1,16)
        WRITE(6,77)IIMAGEB,I,NOT(OUT(I)),NOT(OUT(I)),VOLTS(I)
C77     FORMAT(1H+,4A1,16(Z4,1X))
77      FORMAT(1H+,4A1,I5,1ØX,O6,'       =       ',Z4,1ØX,F6.2)
        ENDDO
        GO TO 3
        END
```

APPENDIX W

[AVA]AUX2

```
        EXTERNAL IO$READVBLK,IO$WRITEVBLK
        BYTE IX(1ØØ),IY(1ØØ),N,IFLAG
        INTEGER*2 D(1ØØ),X,Y,IOSB(4)
        INTEGER*2 US,TS,UM,TM,UH21
        BYTE BD(2ØØ),TEMP
        INTEGER*2 UH84,UH,TH,UD,TD,HD
        CHARACTER*8Ø MSGBUF
        INTEGER*2 TEMS,MS,HMS,TMS
        INTEGER AVAACR
        INTEGER AVACSR,SYS$ASSIGN,SYS$GETMSG,SYS$QIOW,SYS$QIO
        EQUIVALENCE(D,BD)
        ISTATUS=SYS$ASSIGN('AVAØ',IAVAC,,)
        IF(.NOT.ISTATUS)THEN
        TYPE *,' AVA CHANNEL ASSIGN ERROR'
        STOP
        ENDIF
        IFLAG=1
        AVAACR='415'O
        AVACSR=Ø
3       Y=1
        X=Ø
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(IAVAC),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1D,%VAL(36),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
                IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
                ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
                IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
                TYPE *,MSGBUF
        MSGBUF=' '
                ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
                IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
                TYPE *,MSGBUF
                STOP
        ENDIF
        IF(AVACSR.EQ.Ø)AVACSR=1
        DO I=3,35,2
        TEMP=BD(I)
        BD(I)=BD(I+1)
        BD(I+1)=TEMP
        ENDDO
```

[AVA]AUX2

```
      WRITE(6,111)(D(I),I=2,17)
111   FORMAT(1X,16(1X,Z4))
      GO TO 3
      END
```

APPENDIX X

[AVA]AUXPLOT

```
        EXTERNAL IO$READVBLK,IO$WRITEVBLK
        BYTE IX(1ØØ),IY(1ØØ),N,IFLAG
        INTEGER*2 D(26Ø),X,Y,IOSB(4)
        INTEGER*2 OUT(16),US,TS,UM,TM,UH21
        DIMENSION VOLTS(16)
        BYTE BD(52Ø),TEMP,IIMAGEB(4)
        INTEGER*2 UH84,UH,TH,UD,TD,HD
        CHARACTER*8Ø MSGBUF
        INTEGER*2 TEMS,MS,HMS,TMS
        INTEGER AVAACR
        INTEGER AVACSR,SYS$ASSIGN,SYS$GETMSG,SYS$QIOW,SYS$QIO
        EQUIVALENCE(D,BD)
        ISTATUS=SYS$ASSIGN('AVAØ',IAVAC,,)
        IF(.NOT.ISTATUS)THEN
        TYPE *,' AVA CHANNEL ASSIGN ERROR'
        STOP
        ENDIF
        IFLAG=1
        AVAACR='415'O
        AVACSR=Ø
        TYPE *,'ENTER CHANNEL TO BE PLOTTED. (1...16)'
        ACCEPT*,IWCHAN
3       Y=1
        X=Ø
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(IAVAC),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1D,%VAL(52Ø),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
                IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
                ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
                IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
                TYPE *,MSGBUF
        MSGBUF=' '
                ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
                IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
                TYPE *,MSGBUF
                STOP
        ENDIF
        IF(AVACSR.EQ.Ø)AVACSR=1
        DO I=3,519,2
        TEMP=BD(I)
```

[AVA]AUXPLOT

```
        BD(I)=BD(I+1)
        BD(I+1)=TEMP
        ENDDO
C
C       THE FIRST 16 BITS WORD IS TRASH.
C
        IRESET=1
        DO I=2,256,15
C       WRITE(6,111)(D(J),J=I,I+15)
        DO IJ=I,I+15
        ICHAN=IAND(NOT(ISHFT(D(IJ),-12))-1,'F'X)
        IS=IAND(ISHFT(D(IJ),-11),1)
        IF(IS.EQ.Ø)THEN
        II=FLOAT(D(IJ))+1.
        ELSE
        II=D(IJ)
        ENDIF
        OUT(ICHAN+1)=IAND(II,'7777'O)
        VOLTS(ICHAN+1)=FLOAT(IAND(II,'3777'O))/2Ø47.
        VOLTS(ICHAN+1)=VOLTS(ICHAN+1)*1Ø.
        IF(IS.NE.Ø)VOLTS(ICHAN+1)=-VOLTS(ICHAN+1)
        IF(VOLTS(ICHAN+1).GT.Ø.)VOLTS(ICHAN+1)=ABS(VOLTS(ICHAN+1)-1Ø.)
        ENDDO
C       TYPE *,'VOLTS(16)=',VOLTS(16)
        CALL AVT52P(VOLTS,IWCHAN,IRESET)
        ENDDO
        IRESET=1
111     FORMAT(1X,16(1X,Z4))
        GO TO 3
        END
        SUBROUTINE AVT52P(VOLTS,IWCHAN,IRESET)
C
C       THIS IS AUXILIARY DATA VT52 PLOT SUBROUTINE
C
C       IWCHAN IS THE CHANNEL TO BE PLOTTED
C       VOLTS IS THE VOLTAG
C
        BYTE IIMAGEB(4)
        DIMENSION VOLTS(16)
        DATA IFIRST/1/,IXPOS/32/
        IF(IRESET)THEN
        IXPOS=32
        IRESET=Ø
        IFIRST=1
        ENDIF
        IF(IFIRST)THEN
        IFIRST=Ø
        IIMAGEB(1)=27   !ESC
        IIMAGEB(2)=72   !H        CURSOR HOME
        IIMAGEB(3)=27   !ESC
        IIMAGEB(4)=74   !J        ERASE TO END OF SCREEN
        WRITE(6,77)IIMAGEB
77      FORMAT(1H+,4A1)
        IIMAGEB(1)=27   !ESC
        IIMAGEB(2)=89   !Y
```

[AVA]AUXPLOT

```
        IIMAGEB(4)=32   !COLUMN
        IIMAGEB(3)=44   !LINE
        WRITE(6,177)IIMAGEB
177     FORMAT(1H+,4A1,
        1'----------------------------------------------------',
        1'----------------------------')
        IIMAGEB(1)=27   !ESC
        IIMAGEB(2)=89   !Y
        IIMAGEB(4)=32   !COLUMN
        DO I=32,56
        IIMAGEB(3)=I    !LINE
        WRITE(6,178)IIMAGEB
178     FORMAT(1H+,4A1,'+')
        ENDDO
        ENDIF
        IIMAGEB(4)=IXPOS
        Y=24.*((VOLTS(IWCHAN)+10.)/20.)
        IIMAGEB(3)=32.+24.-Y
        WRITE(6,179)IIMAGEB
C       TYPE*,'X,Y=',IIMAGEB(4),IIMAGEB(3)
179     FORMAT(1H+,4A1,'*')
        IXPOS=IXPOS+1
        IF(IXPOS.GE.128)THEN
        IXPOS=32
        IFIRST=1
        ENDIF
        RETURN
        END
```

APPENDIX Y

[AVA]AUXPLOTA

```
      EXTERNAL IO$READVBLK,IO$WRITEVBLK
      BYTE IX(100),IY(100),N,IFLAG
      INTEGER*2 D(260),X,Y,IOSB(4)
      INTEGER*2 OUT(16),US,TS,UM,TM,UH21
      DIMENSION VOLTS(16)
      BYTE BD(520),TEMP,IIMAGEB(4)
      INTEGER*2 UH84,UH,TH,UD,TD,HD
      CHARACTER*80 MSGBUF
      INTEGER*2 TEMS,MS,HMS,TMS
      INTEGER AVAACR
      INTEGER AVACSR,SYS$ASSIGN,SYS$GETMSG,SYS$QIOW,SYS$QIO
      EQUIVALENCE(D,BD)
      ISTATUS=SYS$ASSIGN('AVA0',IAVAC,,)
      IF(.NOT.ISTATUS)THEN
      TYPE *,' AVA CHANNEL ASSIGN ERROR'
      STOP
      ENDIF
      IFLAG=1
      AVAACR='415'O
      AVACSR=0
3     Y=1
      X=0
      ISTATUS=SYS$QIOW(%VAL(1),%VAL(IAVAC),%VAL(%LOC(IO$READVBLK)),
     1IOSB,,,
     1D,%VAL(520),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
              IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
              ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
              IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
              TYPE *,MSGBUF
      MSGBUF=' '
              ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
              IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
              TYPE *,MSGBUF
              STOP
      ENDIF
      IF(AVACSR.EQ.0)AVACSR=1
      DO I=3,519,2
      TEMP=BD(I)
      BD(I)=BD(I+1)
      BD(I+1)=TEMP
```

```
        ENDDO
C
C       THE FIRST 16 BITS WORD IS TRASH.
C
        IRESET=1
        DO I=2,8Ø,15
C       WRITE(6,111)(D(J),J=I,I+15)
        DO IJ=I,I+15
        ICHAN=IAND(NOT(ISHFT(D(IJ),-12))-1,'F'X)
        IS=IAND(ISHFT(D(IJ),-11),1)
        IF(IS.EQ.Ø)THEN
        II=FLOAT(D(IJ))+1.
        ELSE
        II=D(IJ)
        ENDIF
        OUT(ICHAN+1)=IAND(II,'7777'O)
        VOLTS(ICHAN+1)=FLOAT(IAND(II,'3777'O))/2Ø47.
        VOLTS(ICHAN+1)=VOLTS(ICHAN+1)*1Ø.
        IF(IS.NE.Ø)VOLTS(ICHAN+1)=-VOLTS(ICHAN+1)
        IF(VOLTS(ICHAN+1).GT.Ø.)VOLTS(ICHAN+1)=ABS(VOLTS(ICHAN+1)-1Ø.)
        ENDDO
C       TYPE *,'VOLTS(16)=',VOLTS(16)
        CALL AVT52P(VOLTS,IWCHAN,IRESET)
        ENDDO
        IRESET=1
111     FORMAT(1X,16(1X,Z4))
        GO TO 3
        END
        SUBROUTINE AVT52P(VOLTS,IWCHAN,IRESET)
C
C       THIS IS AUXILIARY DATA VT52 PLOT SUBROUTINE
C
C       IWCHAN IS THE CHANNEL TO BE PLOTTED
C       VOLTS IS THE VOLTAG
C
        BYTE IIMAGEB(4)
        DIMENSION VOLTS(16)
        DATA IFIRST/1/,IXPOS/32/
        IF(IRESET)THEN
        IXPOS=32
        IRESET=Ø
        IFIRST=1
        ENDIF
        IF(IFIRST)THEN
        IFIRST=Ø
        IIMAGEB(1)=27    !ESC
        IIMAGEB(2)=72    !H        CURSOR HOME
        IIMAGEB(3)=27    !ESC
        IIMAGEB(4)=74    !J        ERASE TO END OF SCREEN
        WRITE(6,77)IIMAGEB
77      FORMAT(1H+,4A1)
        IIMAGEB(1)=27    !ESC
        IIMAGEB(2)=89    !Y
        IIMAGEB(4)=32    !COLUMN
        IIMAGEB(3)=44    !LINE
```

[AVA]AUXPLOTA

```
        WRITE(6,177)IIMAGEB
177     FORMAT(1H+,4A1,
        1'-----------------------------------------------------',
        1'--------------------')
        IIMAGEB(1)=27    !ESC
        IIMAGEB(2)=89    !Y
        IIMAGEB(4)=32    !COLUMN
        DO I=32,56
        IIMAGEB(3)=I     !LINE
        WRITE(6,178)IIMAGEB
178     FORMAT(1H+,4A1,'+')
        ENDDO
        ENDIF
        DO JJ=1,16
        IIMAGEB(4)=IXPOS
        Y=24.*((VOLTS(JJ)+1Ø.)/2Ø.)
        IIMAGEB(3)=32.+24.-Y
        WRITE(6,179)IIMAGEB
C       TYPE*,'X,Y=',IIMAGEB(4),IIMAGEB(3)
179     FORMAT(1H+,4A1,'*')
        IXPOS=IXPOS+1
        IF(IXPOS.GE.128)THEN
        IXPOS=32
        IFIRST=1
        ENDIF
        ENDDO
        RETURN
        END
```

APPENDIX Z

[AVA]AUXIRIG

```
        EXTERNAL IO$READVBLK,IO$WRITEVBLK
        BYTE IX(100),IY(100),N,IFLAG
        INTEGER*2 D(100),X,Y
        INTEGER*2 US,TS,UM,TM,UH21
        INTEGER*2 UH84,UH,TH,UD,TD,HD
        BYTE IIMAGEB(4)
        INTEGER*2 TEMS,MS,HMS,TMS
        INTEGER AVACSR,SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER AVAACR
        ISTATUS=SYS$ASSIGN('AVA0',IAVAC,,)
        IF(.NOT.ISTATUS)THEN
        TYPE *,' AVA CHANNEL ASSIGN ERROR'
        STOP
        ENDIF
        IFLAG=1
        AVACSR=0
        AVAACR='435'O
        IIMAGEB(1)=27   !ESC
        IIMAGEB(2)=72   !H       CURSOR HOME
        IIMAGEB(3)=27   !ESC
        IIMAGEB(4)=74   !J       ERASE TO END OF SCREEN
        WRITE(6,77)IIMAGEB
        IIMAGEB(1)=27   !ESC
        IIMAGEB(2)=89   !Y
        IIMAGEB(4)=32   !COLUMN
        IIMAGEB(3)=37   !LINE
3       X='1000'O
        Y=2
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(IAVAC),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1D,%VAL(8),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
        IF(AVACSR.EQ.0)AVACSR=1
        DO I=2,4
        D(I)=NOT(D(I))
        ENDDO
        HMS=IAND(ISHFT(D(2),-8),'F'X)
        TMS=IAND(ISHFT(D(2),-4),'F'X)
        MS=IAND(D(2),'F'X)
        US=IAND(ISHFT(D(2),-12),'F'X)
        TS=IAND(D(3),7)
        UM=IAND(ISHFT(D(3),-3),'F'X)
        TM=IAND(ISHFT(D(3),-7),7)
        UH=IAND(ISHFT(D(3),-10),'F'X)
        TH=IAND(ISHFT(D(3),-14),3)
        UD=IAND(D(4),'F'X)
        TD=IAND(ISHFT(D(4),-4),'F'X)
        HD=IAND(ISHFT(D(4),-8),'F'X)

        WRITE(6,13)IIMAGEB,(NOT(D(I)),I=2,4),HD,TD,UD,TH,UH,TM,UM,TS,US,
        1HMS,TMS,MS
77      FORMAT(1H+,4A1,I5,10X,O6,'        =        ',Z4,10X,F6.2)
13      FORMAT(1H+,4A1,3(1X,O6),5X,3Z1,':',1X,2Z1,':',Z1,Z1,':',2Z1,
        1':',3Z1)
        GO TO 3
        END
```

APPENDIX AA

[AVA]AUXIRIG2

```
      EXTERNAL IO$READVBLK,IO$WRITEVBLK
      BYTE IX(100),IY(100),N,IFLAG
      INTEGER*2 D(100),X,Y
      INTEGER*2 US,TS,UM,TM,UH21
      INTEGER*2 UH84,UH,TH,UD,TD,HD
      INTEGER*2 TEMS,MS,HMS,TMS
      INTEGER AVACSR,SYS$ASSIGN,SYS$QIOW,SYS$QIO
      INTEGER AVAACR
      ISTATUS=SYS$ASSIGN('AVA0',IAVAC,,)
      IF(.NOT.ISTATUS)THEN
      TYPE *,' AVA CHANNEL ASSIGN ERROR'
      STOP
      ENDIF
      IFLAG=1
      AVACSR=0
      AVAACR='435'O
3     X='1000'O
      Y=2
      ISTATUS=SYS$QIOW(%VAL(1),%VAL(IAVAC),%VAL(%LOC(IO$READVBLK)),
      1IOSB,,,
      1D,%VAL(8),%VAL(X),%VAL(Y),%VAL(AVACSR),%VAL(AVAACR))
      IF(AVACSR.EQ.0)AVACSR=1
      DO I=2,4
      D(I)=NOT(D(I))
      ENDDO
      HMS=IAND(ISHFT(D(2),-8),'F'X)
      TMS=IAND(ISHFT(D(2),-4),'F'X)
      MS=IAND(D(2),'F'X)
      US=IAND(ISHFT(D(2),-12),'F'X)
      TS=IAND(D(3),7)
      UM=IAND(ISHFT(D(3),-3),'F'X)
      TM=IAND(ISHFT(D(3),-7),7)
      UH=IAND(ISHFT(D(3),-10),'F'X)
      TH=IAND(ISHFT(D(3),-14),3)
      UD=IAND(D(4),'F'X)
      TD=IAND(ISHFT(D(4),-4),'F'X)
      HD=IAND(ISHFT(D(4),-8),'F'X)

      WRITE(6,13)(NOT(D(I)),I=1,4),HD,TD,UD,TH,UH,TM,UM,TS,US,
      1HMS,TMS,MS
```

[AVA]AUXIRIG2

```
13      FORMAT(1X,4(1X,O6),5X,3Z1,':',1X,2Z1,':',Z1,Z1,':',2Z1,
        1':',3Z1)
        GO TO 3
        END
```

APPENDIX AB

[AVA.TAPEDRIVE]IRIGREAD

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM CONTINUOUSLY READS THE IRIG FROM THE SEARCH UNIT
C       AND DISPLAYS IT ON THE TERMINAL.
C
C.......................................................................
C
C       THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C               150001  ! TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C               156400  ! UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C               157000  ! STOP
C               157447  ! THE FILTERS ARE SET TO 1 AND 10000 HZ
C........................................................................
C
C       SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN
        INTEGER*2 INPUT,OUTPUT(4),INIT(5),CONT(5)
        INTEGER*2 US,TS,UM,TM,UH21
        INTEGER*2 UH84,UH,TH,UD,TD,HD
        INTEGER*2 TEMS,MS,HMS,TMS
        CHARACTER *80 MSGBUF
        DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O,'157201'O/
```

```
[AVA.TAPEDRIVE]IRIGREAD

        DATA CONT/'1564Ø3'O,'1564Ø5'O,'1564Ø7'O,'156411'O,
        1'1564ØØ'O/
        ISTATUS=SYS$ASSIGN('ODAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1INIT,%VAL(1Ø),,,,)
        IF(ISTATUS.AND.IOSB(1)) GO TO 1
        TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        TYPE *,'I/O STATUS:',MSGBUF
1       ICONT=1
2       CONTINUE
45      FORMAT(O6)
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1CONT(ICONT),%VAL(2),,,,)
        IF(ISTATUS)      GO TO 5ØØ
        TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        TYPE *,' IOSB(1)=',IOSB(1),'  IOSB(2)=',IOSB(2)
5ØØ     CONTINUE
        IF(ICONT.EQ.5)GO TO 5Ø1
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT(ICONT),%VAL(2),,,,)
        IF(ISTATUS) GO TO 5Ø1
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSBBUG=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        TYPE *,'I/O STATUS:',MSGBUF
        GO TO 2
5Ø1     IF(ICONT.EQ.1)THEN
C       C       IST4=IAND(OUTPUT(1),'ØØØØ1Ø'O)
C       C       IST5=IAND(OUTPUT(1),'ØØØØ2Ø'O)
C       C       IST6=IAND(OUTPUT(1),'ØØØØ4Ø'O)
C       C       IST7=IAND(OUTPUT(1),'ØØØ1ØØ'O)
C       C       IST8=IAND(OUTPUT(1),'ØØØ2ØØ'O)
C       C       IST9=IAND(OUTPUT(1),'ØØØ4ØØ'O)
C       C       IST1Ø=IAND(OUTPUT(1),'ØØ1ØØØ'O)
C       C       IST11=IAND(OUTPUT(1),'ØØ2ØØØ'O)
C       C       IF(IST4.NE.Ø)TYPE*,'START TIME FOUND'
C       C       IF(IST5.NE.Ø)TYPE*,'STOP TIME FOUND'
C       C       IF(IST6.NE.Ø)TYPE*,'PLAYBACK CYCLE BEGAN'
C       C       IF(IST7.NE.Ø)TYPE*,'STOPPED'
C       C       IF(IST8.NE.Ø)TYPE*,'PLAYBACK INTERVAL'
C       C       IF(IST9.NE.Ø)TYPE*,'SEARCHING'
```

```
[AVA.TAPEDRIVE]IRIGREAD

C       C        IF(IST1Ø.NE.Ø)TYPE*,'POWER OFF'
C       C        IF(IST11.NE.Ø)TYPE*,'REMOTE SELECTED'
        ENDIF
        IF(ICONT.LT.5)THEN
        ICONT=ICONT+1
        GO TO 2
        ENDIF
        ICONT=1
C
C       AFTER ALL STATUS AND IRIG HAVE BEEN READ IN LETS PRINT THEM OUT

C
C       IRIG WORD 1 DIGIT DECODING
C
        US=IAND(OUTPUT(2),'F'X)
        TS=IAND(ISHFT(OUTPUT(2),-4),'7'X)
        UM=IAND(ISHFT(OUTPUT(2),-7),'F'X)
        TM=IAND(ISHFT(OUTPUT(2),-11),'7'X)
        UH21=ISHFT(OUTPUT(2),-14)
C
C       IRIG WORD 2 DIGIT DECODING
C
        UH84=ISHFT(IAND(OUTPUT(3),'3'X),2)
        UH=IOR(UH84,UH21)
        TH=IAND(ISHFT(OUTPUT(3),-2),'3'X)
        UD=IAND(ISHFT(OUTPUT(3),-4),'F'X)
        TD=IAND(ISHFT(OUTPUT(3),-8),'F'X)
        HD=ISHFT(OUTPUT(3),-12)
C
C       IRIG WORD 3 DIGIT DECODING
C
        TEMS=IAND(OUTPUT(4),'F'X)
        MS=IAND(ISHFT(OUTPUT(4),-4),'F'X)
        HMS=IAND(ISHFT(OUTPUT(4),-8),'F'X)
        TMS=ISHFT(OUTPUT(4),-12)
        WRITE(6,71)OUTPUT(1),HD,TD,UD,TH,UH,
        1TM,UM,TS,US,TMS,HMS,MS,TEMS
71      FORMAT(1X,'HEX STATUS= ',Z4,4X,'IRIG TIME= ',
        13I1,':',2I1,':',2I1,':',2I1,'.',4I1)
        GO TO 2
        END
```

APPENDIX AC

[AVA.TAPEDRIVE]COMMAND

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM LETS YOU ENTER A SIX DIGIT OCTAL COMMAND (16 BITS)
C       TO THE ON LINE DIGITIZER.
C
C.............................................................................
C
C       THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C               150001  ! TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C               156400  ! UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C               157000  ! STOP
C               157476  ! THE FILTERS ARE SET TO 120 ips I HOPE
C.............................................................................
C
C       SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN
        INTEGER*2 INPUT,OUTPUT,INIT(4)
        CHARACTER *80 MSGBUF
        DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O/
        ISTATUS=SYS$ASSIGN('ODA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1INIT,%VAL(8),,,,)
2       TYPE *,'INPUT CONTROL WORD'
        READ(5,45)INPUT
45      FORMAT(O6)
1       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1INPUT,%VAL(2),,,,)
        IF(ISTATUS)     GO TO 5ØØ
        TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        TYPE *,' IOSB(1)=',IOSB(1),'  IOSB(2)=',IOSB(2)
5ØØ     CONTINUE
11      FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIOW IO-STATUS RETURN:',MSGBUF
        WRITE(6,11)INPUT,IOSB(1),IOSB(2),IOSB(3),IOSB(4)
        GO TO 2
        END
```

APPENDIX AD

[AVA.TAPEDRIVE]REVIEW

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM CONTINUOUSLY READS THE IRIG FROM THE SEARCH UNIT
C       BUT DISPLAYS ONLY THE DESIRED "SAVE" IRIG DESIGNATED BY THE
C       USER WHEN HE HITS THE RETURN KEY.
C
C       THE "SAVE" IRIGS ARE
C       WRITTEN TO DISK IN REVIEW.IRG AS THEY ARE COLLECTED.
C
C
C.......................................................................
C
C       THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C               150001  ! TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C               156400  ! UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C               157000  ! STOP
C               157447  ! THE FILTERS ARE SET TO 1 AND 10000 HZ
C........................................................................
C
C       SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN
        INTEGER*2 INPUT,OUTPUT(4),INIT(5),CONT(5)
```

[AVA.TAPEDRIVE]REVIEW

```
         INTEGER*2 US,TS,UM,TM,UH21
         INTEGER*2 UH84,UH,TH,UD,TD,HD
         INTEGER*2 TEMS,MS,HMS,TMS
         CHARACTER *80 MSGBUF,GETIRIG*1
         DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O,'157201'O/
         DATA CONT/'156403'O,'156405'O,'156407'O,'156411'O,
         1'156400'O/
         OPEN(UNIT=8,NAME='REVIEW.IRG',TYPE='NEW')
         ISTATUS=SYS$ASSIGN('ODA0',ITCHAN,,)
         IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
         ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
         1IOSB,,,
         1INIT,%VAL(10),,,,)
         IF(ISTATUS.AND.IOSB(1)) GO TO 1
         TYPE *,' ERROR IN QIOW CALL'
         ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
         IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
         TYPE *,'QIO PARAMETER STATUS:',MSGBUF
         MSGBUF=' '
         ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
         TYPE *,'I/O STATUS:',MSGBUF
1        ICONT=1
         TYPE *,' TO SAVE IRIG HIT RETURN WHEN DESIRED SCENE APPEARS.'
         READ(5,5)GETIRIG
5        FORMAT(A)
2        CONTINUE
45       FORMAT(O6)
         ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
         1IOSB,,,
         1CONT(ICONT),%VAL(2),,,,)
         IF(ISTATUS)     GO TO 500
         TYPE *,' ERROR IN QIOW CALL'
         ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
         IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
         TYPE *,'QIO PARAMETER STATUS:',MSGBUF
         TYPE *,' IOSB(1)=',IOSB(1),'  IOSB(2)=',IOSB(2)
500      CONTINUE
         IF(ICONT.EQ.5)GO TO 501
         ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
         1IOSB,,,
         1OUTPUT(ICONT),%VAL(2),,,,)
         IF(ISTATUS) GO TO 501
         ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
         IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
         TYPE *,'QIO PARAMETER STATUS:',MSGBUF
         MSBBUG=' '
         ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
         TYPE *,'I/O STATUS:',MSGBUF
         GO TO 2
501      IF(ICONT.EQ.1)THEN
C        C       IST4=IAND(OUTPUT(1),'000010'O)
C        C       IST5=IAND(OUTPUT(1),'000020'O)
C        C       IST6=IAND(OUTPUT(1),'000040'O)
C        C       IST7=IAND(OUTPUT(1),'000100'O)
C        C       IST8=IAND(OUTPUT(1),'000200'O)
```

```
C       C       IST9=IAND(OUTPUT(1),'ØØØ4ØØ'O)
C       C       IST1Ø=IAND(OUTPUT(1),'ØØ1ØØØ'O)
C       C       IST11=IAND(OUTPUT(1),'ØØ2ØØØ'O)
C       C       IF(IST4.NE.Ø)TYPE*,'START TIME FOUND'
C       C       IF(IST5.NE.Ø)TYPE*,'STOP TIME FOUND'
C       C       IF(IST6.NE.Ø)TYPE*,'PLAYBACK CYCLE BEGAN'
C       C       IF(IST7.NE.Ø)TYPE*,'STOPPED'
C       C       IF(IST8.NE.Ø)TYPE*,'PLAYBACK INTERVAL'
C       C       IF(IST9.NE.Ø)TYPE*,'SEARCHING'
C       C       IF(IST1Ø.NE.Ø)TYPE*,'POWER OFF'
C       C       IF(IST11.NE.Ø)TYPE*,'REMOTE SELECTED'
        ENDIF
        IF(ICONT.LT.5)THEN
        ICONT=ICONT+1
        GO TO 2
        ENDIF
        ICONT=1
C
C       AFTER ALL STATUS AND IRIG HAVE BEEN READ IN LETS PRINT THEM OUT

C
C       IRIG WORD 1 DIGIT DECODING
C
        US=IAND(OUTPUT(2),'F'X)
        TS=IAND(ISHFT(OUTPUT(2),-4),'7'X)
        UM=IAND(ISHFT(OUTPUT(2),-7),'F'X)
        TM=IAND(ISHFT(OUTPUT(2),-11),'7'X)
        UH21=ISHFT(OUTPUT(2),-14)
C
C       IRIG WORD 2 DIGIT DECODING
C
        UH84=ISHFT(IAND(OUTPUT(3),'3'X),2)
        UH=IOR(UH84,UH21)
        TH=IAND(ISHFT(OUTPUT(3),-2),'3'X)
        UD=IAND(ISHFT(OUTPUT(3),-4),'F'X)
        TD=IAND(ISHFT(OUTPUT(3),-8),'F'X)
        HD=ISHFT(OUTPUT(3),-12)
C
C       IRIG WORD 3 DIGIT DECODING
C
        TEMS=IAND(OUTPUT(4),'F'X)
        MS=IAND(ISHFT(OUTPUT(4),-4),'F'X)
        HMS=IAND(ISHFT(OUTPUT(4),-8),'F'X)
        TMS=ISHFT(OUTPUT(4),-12)
        WRITE(6,71)OUTPUT(1),HD,TD,UD,TH,UH,
        1TM,UM,TS,US,TMS,HMS,MS,TEMS
        WRITE(8,71)OUTPUT(1),HD,TD,UD,TH,UH,
        1TM,UM,TS,US,TMS,HMS,MS,TEMS
71      FORMAT(1X,'HEX STATUS= ',Z4,4X,'IRIG TIME= ',
        13I1,':',2I1,':',2I1,':',2I1,'.',4I1)
        READ(5,5,END=7777)GETIRIG
        GO TO 2
7777    CLOSE(UNIT=8)
        STOP ' REVIEW.IRG GENERATED'
        END
```

APPENDIX AE

[AVA.TAPEDRIVE]RTODISK

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C        THIS PROGRAM USES THE REVIEW.IRG FILE TO FIND SEARCH TIMES.
C
C        THE FOLLOWING INSTRUCTIONS ARE SENT TO THE IRIG SEARCH UNIT
C        ON THE ON LINE DIGITIZER AND THEN THE REVIEW.IRG FILE IS READ FOR THE
C        FIRST SEARCH TIME.
C
C        150001  ! TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C        156400  ! UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C        157000  ! STOP
C        157447  ! THE FILTERS ARE SET TO 120 ips I HOPE, NO CARRIER FILTER
C
C        ENTER IRIG INPUT:
C
C        THE FOLLOWING IS THE SEQUENCE OF CONTROL WORDS THAT WOULD BE SENT
C        IF THE OPERATOR ENTERS A START IRIG TIME OF 000:00:01:00.0000
C
C        150400  ! SEARCH START TIME DAYS "00X" WHERE X IS NOT SET IN THIS WORD
C        151000  ! SEARCH START TIME DAYS "SS0", HOURS "00" WHERE SS WAS SET ABOVE
C        151401  ! SEARCH START TIME MINUTES "01" MINUTES
C        152000  ! SEARCH START TIME SECONDS "00" SECONDS
C        152400  ! SEARCH START TIME MILLISECONDS .00XX
C        153000  ! SEARCH START TIME MILLISECONDS .SS00 WHERE SS WAS SET ABOVE
C
C        THE PROGRAM AFTER THE IRIG IS ENTERED TRANFERS TO THE IRIG SEARCH
C        UNIT TRANSFERS THE FOLLOWING CONTROL WORD AND THE SEARCH PROCESS IS
C        INITIATED.
C
C        157227  ! SEARCH TO START TIME  000:00:01:00.0000 OR USER ENTERED
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         EXTERNAL IO$WRITEVBLK,IO$READVBLK
         INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
         INTEGER SYS$GETMSG,SYS$BINTIM,SYS$SETIMR,SYS$WAITFR
         INTEGER*2 IOSB(4),MSGLEN,DSTATUS
         INTEGER*2 READCNTRL(3)
         INTEGER*2 INPUT(11),OUTPUT(4)
         INTEGER*2 SW1,SW2,SW3,SW4,SW5,SW6,SW7,SW8,SW9,SW10,SW11,SW12
```

[AVA.TAPEDRIVE]RTODISK

```
          DOUBLE PRECISION QUAD
          CHARACTER*16 TIME,FILENAME*6Ø
          INTEGER*2 TM,UM,TS,US,ITMS,HMS,TMS
          INTEGER*2 HD,ITD,TD,UD,UH,TH
          INTEGER*2 IDAY,IHR,IMIN,ISEC,IMSEC
          INTEGER*2 TEMS,IUMS,UMS
          CHARACTER *8Ø MSGBUF
          EQUIVALENCE(SW1,INPUT(5)),(SW2,INPUT(6)),(SW3,INPUT(7))
          EQUIVALENCE(SW4,INPUT(8)),(SW5,INPUT(9)),(SW6,INPUT(1Ø))
          COMMON/AVACHAN/IAVACHAN
          DATA INPUT/'15ØØØ1'O,'1564ØØ'O,'157ØØØ'O,'157447'O,'15Ø4ØØ'O,
          1'151ØØØ'O,'1514Ø1'O,'152ØØØ'O,'1524ØØ'O,'153ØØØ'O,'157227'O/
          DATA READCNTRL/'157442'O,'1572Ø1'O,'157Ø61'O/
          ISTATUS=SYS$ASSIGN('AVAØ',IAVACHAN,,)
          IF(.NOT.ISTATUS)TYPE *,' ERROR IN AVA CHANNEL ASSIGN'
          ISTATUS=SYS$ASSIGN('ODAØ',ITCHAN,,)
          IF(.NOT.ISTATUS)TYPE *,' ERROR IN ON LINE DIGITIZER CHANNEL ASSIGN'
          TYPE *,'ENTER BEGINNING FILE NAME TO BE USED FOR DISK FILES',
          1'. (i.e. XØØ32ØØØØ)'
          READ(5,45)FILENAME
45        FORMAT(A)
C2        TYPE *,'ENTER SEARCH START IRIG TIME IN THE FOLLOWING FORMAT:'
C         TYPE *,'DAY:HR:MN:SC.MSEC'
C         TYPE *,'ØØØ:ØØ:Ø1:ØØ.ØØØØ FOR EXAMPLE'
C         READ(5,45)IDAY,IHR,IMIN,ISEC,IMSEC
C45       FORMAT(I3,1X,I2,1X,I2,1X,I2,1X,I4)
          OPEN(UNIT=8,NAME='REVIEW.IRG',STATUS='OLD')
2         READ(8,71,END=7777)OUTPUT(1),HD,TD,UD,TH,UH,
          1TM,UM,TS,US,TMS,HMS,MS,TEMS
71        FORMAT(13X,Z4,15X,
          13I1,1X,2I1,1X,2I1,1X,2I1,1X,4I1)
C         HD=IDAY/1ØØ
C         ITD=IDAY-(HD*1ØØ)
C         TD=ITD/1Ø
C         UD=ITD-(TD*1Ø)
          SW1='ØØ642Ø'O
          SW1=ISHFT(IOR(SW1,TD),4)
          SW1=IOR(SW1,UD)
          SW2='ØØ151Ø'O
          SW2=ISHFT(IOR(IAND(HD,3),SW2),2)
C         TH=IHR/1Ø
          SW2=ISHFT(IOR(IAND(TH,3),SW2),4)
C         UH=IHR-(TH*1Ø)
          SW2=IOR(SW2,UH)
          SW3='ØØ646Ø'O
C         TM=IMIN/1Ø
C         UM=IMIN-(TM*1Ø)
          SW3=ISHFT(IOR(SW3,TM),4)
          SW3=IOR(SW3,UM)
C         TS=ISEC/1Ø
C         US=ISEC-(TS*1Ø)
          SW4='ØØ65ØØ'O
          SW4=ISHFT(IOR(SW4,TS),4)
          SW4=IOR(SW4,US)
          SW5='ØØ652Ø'O
```

```
        SW6='ØØ654Ø'O
C       HMS=IMSEC/1ØØØ
C       ITMS=IMSEC-(HMS*1ØØØ)
C       TMS=ITMS/1ØØ
C       IUMS=ITMS-(TMS*1ØØ)
C       UMS=IUMS/1Ø
C       TEMS=IUMS-(UMS*1Ø)
        SW5=ISHFT(IOR(SW5,HMS),4)
        SW5=IOR(SW5,TMS)
        SW6=ISHFT(IOR(SW6,UMS),4)
        SW6=IOR(SW6,TEMS)
C       WRITE(6,55) SW1,SW2,SW3,SW4,SW5,SW6
55      FORMAT(1X,'SWX=',6(1X,Z4))
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1INPUT,%VAL(22),,,,)

        WRITE(6,171)OUTPUT(1),HD,TD,UD,TH,UH,
        1TM,UM,TS,US,TMS,HMS,MS,TEMS
171     FORMAT(1X,'SEARCHING FOR HEX STATUS= ',Z4,4X,'IRIG TIME= ',
        13I1,':',2I1,':',2I1,':',2I1,'.',4I1)
        IF(.NOT.ISTATUS.OR..NOT. IOSB(1))GO TO 34
C       CHECK TAPEDRIVE STATUS

61      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1'1564Ø3'O,%VAL(2),,,,) !TELL DATUM YOU WANT STATUS
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1DSTATUS,%VAL(2),,,,)
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1'1564ØØ'O,%VAL(2),,,,) !CLEAR DATUM
        IF(IAND(DSTATUS,'1ØØ'O).EQ.Ø)GO TO 61    !IS IT STOPPED??
C       MOVE FOWARD TO CORRECT IRIG THEN PLAY TAPE AT 32/1 AND TRANSFER IMAGES.
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1READCNTRL,%VAL(4),,,,) !SET UP PLAYBACK FILTER AND MOVE BACK FOWARD 12Ø IPS
        TIME='ØØØØ ØØ:ØØ:Ø9.6Ø'
        ISTATUS=SYS$BINTIM(%DESCR(TIME),QUAD)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        ISTATUS=SYS$SETIMR(%VAL(6),QUAD,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        ISTATUS=SYS$WAITFR(%VAL(6))
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
C       NOW PLAYBACK AT 32/1
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1READCNTRL(3),%VAL(2),,,,)        !SET UP PLAYBACK FILTER AND MOVE BACK FOWARD 12Ø IPS
62      ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1'1564Ø3'O,%VAL(2),,,,) !TELL DATUM YOU WANT STATUS
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1DSTATUS,%VAL(2),,,,)
```

[AVA.TAPEDRIVE]RTODISK

```
        IF(IAND(DSTATUS,2).EQ.Ø)GO TO 62         !IS TAPE SYNC ON?
C
C       GIVE AVA TIME TO LOAD ALL FIELDS
C
        TIME='ØØØØ ØØ:ØØ:Ø3.ØØ'
        ISTATUS=SYS$BINTIM(%DESCR(TIME),QUAD)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        ISTATUS=SYS$SETIMR(%VAL(6),QUAD,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
        ISTATUS=SYS$WAITFR(%VAL(6))
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN TIME DELAY'
C
C       IS FIELD 1 BEING LOADED?  (Ø , 1 , 2 , 3)
C
23      CALL FIELD(IFIELD,IAVACSR)
        IF(IFIELD.NE.1)GO TO 23

C
C       WRITE THE IMAGE TO DISK
C
        CALL AVATODSK2(FILENAME)
        GO TO 2
34      TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        TYPE *,'I/O STATUS:',MSGBUF
5ØØ     CONTINUE
11      FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
        GO TO 2
7777    ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1'15722Ø'O,%VAL(2),,,,) !STOP HBR-3ØØØ TAPE DRIVE
        STOP 'ALL IRIGS HAVE BEEN FOUND'
        END
        SUBROUTINE FIELD(IFIELD,AVACSR)
        INTEGER AVACSR
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN,NPUT,X,Y
        INTEGER*2 INPUT,OUTPUT,INIT(4)
        CHARACTER *8Ø MSGBUF
        COMMON/AVACHAN/ITCHAN
        IAVACSR='4ØØ1'O !SET MEMORY WINDOW ENABLE AND INITIALIZE AVA
        ISAVE=AVACSR
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT,%VAL(2),%VAL(X),%VAL(Y),%VAL(IAVACSR),%VAL(IAVAACR))
        IF(ISTATUS)     GO TO 5Ø1
        TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
```

[AVA.TAPEDRIVE]RTODISK

```
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
5Ø1     AVACSR=ISAVE
        IFIELD=IAND(OUTPUT,3)
        RETURN
        END
```

APPENDIX AF

[AVA.TAPEDRIVE]STOSTART

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       THIS PROGRAM LETS THE OPERATOR ENTER THE START SEARCH TIME FROM THE
C       KEYBOARD AND THEN TELLS THE SEARCH UNIT TO GO AND SEARCH FOR THE IRIG
C       TIME ENTERED.
C
C       THE FOLLOWING INSTRUCTIONS ARE SENT TO THE IRIG SEARCH UNIT
C       ON THE ON LINE DIGITIZER AND THEN THE OPERATOR IS ASKED FOR START IRIG.
C
C       150001  ! TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C       156400  ! UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C       157000  ! STOP
C       157447  ! THE FILTERS ARE SET TO 120 fps I HOPE, NO CARRIER FILTER
C
C       ENTER IRIG INPUT:
C
C       THE FOLLOWING IS THE SEQUENCE OF CONTROL WORDS THAT WOULD BE SENT
C       IF THE OPERATOR ENTERS A START IRIG TIME OF 000:00:01:00.0000
C
C       150400  ! SEARCH START TIME DAYS "00X" WHERE X IS NOT SET IN THIS WORD
C       151000  ! SEARCH START TIME DAYS "SS0", HOURS "00" WHERE SS WAS SET ABOVE
C       151401  ! SEARCH START TIME MINUTES "01" MINUTES
C       152000  ! SEARCH START TIME SECONDS "00" SECONDS
C       152400  ! SEARCH START TIME MILLISECONDS .00XX
C       153000  ! SEARCH START TIME MILLISECONDS .SS00 WHERE SS WAS SET ABOVE
C
C       THE PROGRAM AFTER THE IRIG IS ENTERED TRANFERS TO THE IRIG SEARCH
C       UNIT TRANSFERS THE FOLLOWING CONTROL WORD AND THE SEARCH PROCESS IS
C       INITIATED.
C
C       157207  ! SEARCH TO START TIME  000:00:01:00.0000 OR USER ENTERED
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN
        INTEGER*2 INPUT(11)
        INTEGER*2 SW1,SW2,SW3,SW4,SW5,SW6,SW7,SW8,SW9,SW10,SW11,SW12
```

[AVA.TAPEDRIVE]STOSTART

```
        INTEGER*2 TM,UM,TS,US,ITMS,HMS,TMS
        INTEGER*2 HD,ITD,TD,UD,UH,TH
        INTEGER*2 IDAY,IHR,IMIN,ISEC,IMSEC
        INTEGER*2 TEMS,IUMS,UMS
        CHARACTER *8Ø MSGBUF
        EQUIVALENCE(SW1,INPUT(5)),(SW2,INPUT(6)),(SW3,INPUT(7))
        EQUIVALENCE(SW4,INPUT(8)),(SW5,INPUT(9)),(SW6,INPUT(1Ø))
        DATA INPUT/'15ØØØ1'O,'1564ØØ'O,'157ØØØ'O,'157447'O,'15Ø4ØØ'O,
        1'151ØØØ'O,'1514Ø1'O,'152ØØØ'O,'1524ØØ'O,'153ØØØ'O,'1572Ø7'O/
        ISTATUS=SYS$ASSIGN('ODAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN ON LINE DIGITIZER CHANNEL ASSIGN'
2       TYPE *,'ENTER SEARCH START IRIG TIME IN THE FOLLOWING FORMAT:'
        TYPE *,'DAY:HR:MN:SC.MSEC'
        TYPE *,'ØØØ:ØØ:Ø1:ØØ.ØØØØ FOR EXAMPLE'
        READ(5,45)IDAY,IHR,IMIN,ISEC,IMSEC
45      FORMAT(I3,1X,I2,1X,I2,1X,I2,1X,I4)
        HD=IDAY/1ØØ
        ITD=IDAY-(HD*1ØØ)
        TD=ITD/1Ø
        UD=ITD-(TD*1Ø)
        SW1='ØØ642Ø'O
        SW1=ISHFT(IOR(SW1,TD),4)
        SW1=IOR(SW1,UD)
        SW2='ØØ151Ø'O
        SW2=ISHFT(IOR(IAND(HD,3),SW2),2)
        TH=IHR/1Ø
        SW2=ISHFT(IOR(IAND(TH,3),SW2),4)
        UH=IHR-(TH*1Ø)
        SW2=IOR(SW2,UH)
        SW3='ØØ646Ø'O
        TM=IMIN/1Ø
        UM=IMIN-(TM*1Ø)
        SW3=ISHFT(IOR(SW3,TM),4)
        SW3=IOR(SW3,UM)
        TS=ISEC/1Ø
        US=ISEC-(TS*1Ø)
        SW4='ØØ65ØØ'O
        SW4=ISHFT(IOR(SW4,TS),4)
        SW4=IOR(SW4,US)
        SW5='ØØ652Ø'O
        SW6='ØØ654Ø'O
        HMS=IMSEC/1ØØØ
        ITMS=IMSEC-(HMS*1ØØØ)
        TMS=ITMS/1ØØ
        IUMS=ITMS-(TMS*1ØØ)
        UMS=IUMS/1Ø
        TEMS=IUMS-(UMS*1Ø)
        SW5=ISHFT(IOR(SW5,HMS),4)
        SW5=IOR(SW5,TMS)
        SW6=ISHFT(IOR(SW6,UMS),4)
        SW6=IOR(SW6,TEMS)
C       WRITE(6,55) SW1,SW2,SW3,SW4,SW5,SW6
55      FORMAT(1X,'SWX=',6(1X,Z4))
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1INPUT,%VAL(22),,,,)
        IF(.NOT.ISTATUS.OR..NOT. IOSB(1))GO TO 34
        GO TO 2
34      TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        TYPE *,'I/O STATUS:',MSGBUF
500     CONTINUE
11      FORMAT(1X,'INPUT=',O6,2X,'IOSB=',O6,2X,O6,2X,O6,2X,O6)
        GO TO 2
        END
```

APPENDIX AG

[AVA.TAPEDRIVE]STATUSR

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM CONTINUOUSLY READS THE STATUS FROM THE SEARCH UNIT
C       AND DISPLAYS IT ON THE TERMINAL.
C
C..........................................................................
C
C       THE INITIALIZATION SEQUENCE USED IN THIS PROGRAM IS AS FOLLOWS
C
C               150001  ! TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C               156400  ! UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT
C               157000  ! STOP
C               157447  ! THE FILTERS ARE SET TO 1 AND 10000 HZ
C...........................................................................
C
C       SOME TYPICAL COMMANDS ARE AS FOLLOWS:
C
C       150001 = TRANSLATE IRIG A WITH ZERO FRAME BYPASS
C
C       156400 = UPDATE TIME, RESET RECORD ENABLE, RESET INTERRUPT ENABLE
C
C       157201 = DRIVE FORWARD AT 120 ips (NORMAL REALTIME PLAYBACK)
C       157221 = DRIVE FORWARD AT 240 ips (EQUIVALENT TO FAST FORWARD)
C       157061 = DRIVE FORWARD AT 3 3/4 (32 TO 1 PLAYBACK)
C       157000 = STOP
C       157222 = DRIVE REVERSE AT 240 ips
C       157202 = DRIVE REVERSE AT 120 ips
C       157206 = SINGLE CYCLE SEARCH MODE AT 120 ips
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN
        INTEGER*2 INPUT,OUTPUT(4),INIT(5),CONT(5)
        INTEGER*2 US,TS,UM,TM,UH21
        INTEGER*2 UH84,UH,TH,UD,TD,HD
        INTEGER*2 TEMS,MS,HMS,TMS
        CHARACTER *80 MSGBUF
        DATA INIT/'150001'O,'156400'O,'157000'O,'157447'O,'157201'O/
```

[AVA.TAPEDRIVE]STATUSR

```
        DATA CONT/'156403'O,'156405'O,'156407'O,'156411'O,
        1'156400'O/
        ISTATUS=SYS$ASSIGN('ODA0',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
C       ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
C       1IOSB,,,
C       1INIT,%VAL(8),,,,)
C       IF(ISTATUS.AND.IOSB(1)) GO TO 1
C       TYPE *,' ERROR IN QIOW CALL'
C       ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIO PARAMETER STATUS:',MSGBUF
C       MSGBUF=' '
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       TYPE *,'I/O STATUS:',MSGBUF
1       ICONT=1
2       CONTINUE
45      FORMAT(O6)
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1CONT(ICONT),%VAL(2),,,,)
        IF(ISTATUS)     GO TO 500
        TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        TYPE *,' IOSB(1)=',IOSB(1),'  IOSB(2)=',IOSB(2)
500     CONTINUE
        ISTATUS=SYS$QIOW(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1OUTPUT(ICONT),%VAL(2),,,,)
        IF(ISTATUS) GO TO 501
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        MSBBUG=' '
        ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        TYPE *,'I/O STATUS:',MSGBUF
        GO TO 2
501             IST4=IAND(OUTPUT(1),'000010'O)
                IST5=IAND(OUTPUT(1),'000020'O)
                IST6=IAND(OUTPUT(1),'000040'O)
                IST7=IAND(OUTPUT(1),'000100'O)
                IST8=IAND(OUTPUT(1),'000200'O)
                IST9=IAND(OUTPUT(1),'000400'O)
                IST10=IAND(OUTPUT(1),'001000'O)
                IST11=IAND(OUTPUT(1),'002000'O)
                IST1= IAND(OUTPUT(1),'000002'O)
        TYPE *,'.......STATUS IS AS FOLLOWS...............',
        1'............'
                IF(IST1.NE.0)TYPE*,'SYNC'
                IF(IST4.NE.0)TYPE*,'START TIME FOUND'
                IF(IST5.NE.0)TYPE*,'STOP TIME FOUND'
                IF(IST6.NE.0)TYPE*,'PLAYBACK CYCLE BEGAN'
                IF(IST7.NE.0)TYPE*,'STOPPED'
```

[AVA.TAPEDRIVE]STATUSR

```
                IF(IST8.NE.Ø)TYPE*,'PLAYBACK INTERVAL'
                IF(IST9.NE.Ø)TYPE*,'SEARCHING'
                IF(IST1Ø.NE.Ø)TYPE*,'POWER OFF'
                IF(IST11.NE.Ø)TYPE*,'REMOTE SELECTED'
C       WRITE(6,7)OUTPUT(1)
7       FORMAT(1X,O6)
        GO TO 2
        END
```

APPENDIX AH

[AVA.TAPEDRIVE]IOTEST

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS PROGRAM IS USED TO TEST THE ON LINE DIGITIZER DR11-C INTERFACE
C       IN CONJUNCTION WITH THE ODDRIVER. THE MAINTENENCE CABLE MUST BE HOOKED
C       UP!
C
C       THE ODDRIVER TRANSFERS THE WORD IN INPUT TO THE OUTBUF REGISTER ON
C       THE DR11-C. THE DRIVER THEN SETS CSRØ AND THE IEA BITS. THIS WILL
C       CAUSE AN INTERRUPT AFTER IPL IS LOWERED BELOW DEVICE IPL. AFTER
C       THE INTERRUPT THE INPUTBUF IS THE COPIED INTO IOSB(3) AND IS USED IN
C       THIS PROGRAM TO COMPARE TO WHAT WENT INTO THE OUTBUF.
C
C       THIS PROGRAM TESTS ALL DATA BITS ON THE DR11-C AND THE "A" INTERRUPT
C       HARDWARE.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        EXTERNAL IO$WRITEVBLK,IO$READVBLK
        INTEGER SYS$ASSIGN,SYS$QIOW,SYS$QIO
        INTEGER SYS$GETMSG
        INTEGER*2 IOSB(4),MSGLEN
        INTEGER*2 INPUT,OUTPUT
        CHARACTER *8Ø MSGBUF
        ISTATUS=SYS$ASSIGN('ODAØ',ITCHAN,,)
        IF(.NOT.ISTATUS)TYPE *,' ERROR IN DR11-C CHANNEL ASSIGN'
        INPUT='1ØØØØØ'O
2       TYPE *,'DR11-C BIT TEST STARTING..........'
1       ISTATUS=SYS$QIO(%VAL(1),%VAL(ITCHAN),%VAL(%LOC(IO$WRITEVBLK)),
        1IOSB,,,
        1INPUT,%VAL(2),,,,)
        IF(ISTATUS)     GO TO 5ØØ
        TYPE *,' ERROR IN QIOW CALL'
        ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO PARAMETER STATUS:',MSGBUF
        TYPE *,' IOSB(1)=',IOSB(1),'  IOSB(2)=',IOSB(2)
5ØØ     CONTINUE
11      FORMAT(1X,O6,2X,O6,'CSR=',O6,2X,O6)
C       ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C       IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C       TYPE *,'QIOW IO-STATUS RETURN:',MSGBUF
        IF(INPUT.NE.IOSB(3))THEN
        TYPE *,' INBUF NOT EQUAL TO OUTBUF'
        WRITE(6,11)IOSB(1),IOSB(2),IOSB(3),IOSB(4)
        WRITE(6,12) INPUT,IOSB(3)
12      FORMAT(1X,'OUTPUT WAS=',O6,5X,'INPUT WAS=',O6)
        ENDIF
        INPUT=INPUT+1
        IF(INPUT.GE.32767)THEN
        INPUT='1ØØØØØ'O
        TYPE *,'DR11-C BIT TEST COMPLETED'
        GO TO 2
        ENDIF
        GO TO 1
        END
```

APPENDIX AI

AVA FRAME BUFFER I/O DRIVER

```
        .TITLE  AVDRIVER - VAX/VMS AVA FRAME BUFFER INTERFACE DRIVER
        .IDENT  'V03-001'
;++
;
; FACILITY:
;
;       VAX/VMS ON LINE DIGITIZER AVA FRAME BUFFER
;
; ABSTRACT:
;
;       This module contains the driver:
;
;               Tables
;               Controller and unit initialization routines
;               The FDT routine
;               The start I/O routine
;               The interrupt service routine
;               The cancel I/O routine
;               The device register dump routine
;
; AUTHOR:
;
;       S. Richard F. Sims  January 27,1983
;
; REVISION HISTORY:
;
;--
        .SBTTL  External and local symbol definitions
;
; External symbols
;
        $CANDEF                         ; Cancel reason codes
        $CRBDEF                         ; Channel request block
        $DCDEF                          ; Device classes and types
        $DDBDEF                         ; Device data block
        $DEVDEF                         ; Device characteristics
        $IDBDEF                         ; Interrupt data block
        $IODEF                          ; I/O function codes
        $IPLDEF                         ; Hardware IPL definitions
        $IRPDEF                         ; I/O request packet
```

AVA FRAME BUFFER I/O DRIVER

```
        $SSDEF                          ; System status codes
        $UCBDEF                         ; Unit control block
        $VECDEF                         ; Interrupt vector block
        $JIBDEF                         ; JOB INFO BLOCK OFFSET DEFS
        $PCBDEF                         ; PROCESS CONTROL BLOCK OFFSET DEFS
;
; Local symbols
;
;
; Argument list (AP) offsets for device-dependent QIO parameters
;
P1      = Ø                             ; First QIO parameter
P2      = 4                             ; Second QIO parameter
P3      = 8                             ; Third QIO parameter
P4      = 12                            ; Fourth QIO parameter
P5      = 16                            ; Fifth QIO parameter
P6      = 2Ø                            ; Sixth QIO parameter
;
; Other constants
;
AVDEFBUFSIZ     = 1                     ; Default buffer size
AVTIMEOUTSEC    = 5                     ; 1Ø second device timeout
AVNUMREGS       = 3                     ; Device has 3 registers
BUFOVRHD        = 12                    ; SYSTEM BUFFER OVERHEAD FOR BUFFERED I/O
;
; Definitions that follow the standard UCB fields
;
        $DEFINI UCB                     ; Start of UCB definitions
        .=UCB$K_LENGTH                  ; Position at end of UCB
$DEF    UCB$W_AVCSR                     ; UCB Device CSR STORAGE
                        .BLKW   1
$DEF    UCB$W_AVBYTCNT                  ; Device's BYTE count register
                        .BLKW   1
$DEF    UCB$W_AVOUTBUF                  ; DEVICE OUTBUF REGISTER
                        .BLKW   1
$DEF    UCB$W_AVXADDR                   ; STARTING X ADDRESS (P3)
                        .BLKW   1
$DEF    UCB$W_AVYADDR                   ; STARTING Y ADDRESS (P4)
                        .BLKW   1
$DEF    UCB$W_AVACR                     ; INITIALIZE ACCES CONTROL REGISTER BITS
                        .BLKW   1
$DEF    UCB$K_AVUCBLEN                  ; Length of extended UCB
                        .BLKW   1
;
; Bit positions for device-dependent status field in UCB
;
        $VIELD  UCBCSR,Ø,<-             ; Device status
                <BITZERO,,M>,-          ; First bit
                <BITONE,,M>,-           ; Second bit
                >
        $DEFEND UCB                     ; End of UCB definitions
;
; Device register offsets from CSR address
;
        $DEFINI AV                      ; Start of status definitions
```

```
AVA FRAME BUFFER I/O DRIVER

$DEF    AVCSR                         ; Control/status
                    .BLKW   1
;
; Bit positions for device control/status register
;
        VIELD   AVCSR,Ø,<-            ; Control/status register
                <AVEN,,M>,-           ; WHEN UNASSERTED INITIALIZES AVA HARDWARE
                <INTENØ,,M>,-         ; ENABLES AVA INTERRUPT Ø
                <INTEN1,,M>,-         ; ENABLES AVA INTERRUPT 1
                <INTEN2,,M>,-         ; ENABLES AVA INTERRUPT 2
                <INTEN3,,M>,-         ; ENABLES AVA INTERRUPT 3
                <DMAEN,,M>,-          ; ENABLES AVA UNIBUS MASTERSHIP REQUEST
                <OTAGØ,,M>,-          ; GENERAL PURPOSE SENSE LINE
                <ITAG,,M>,-           ; GENERAL PURPOSE SENSE LINE
                <WDRØ,,M>,-           ; MEMORY WINDOW ID BIT Ø
                <WDR1,,M>,-           ; MEMORY WINDOW ID BIT 1
                <WDR2,,M>,-           ; MEMORY WINDOW ID BIT 2
                <WDEN,,M>,-           ; ENABLES AVA RESPONSE TO MEMORY WINDOW ACCESS
                <OTAG1,,M>,-          ; GENERAL PURPOSE SOFTWARE TAG
                <,3>,-                ; RESERVED FOR FURTURE USE
        >
        $DEFEND AV                    ; End of device register
                                      ; definitions.
                FSMCCSR=        AVCSR-OØØØ176   ; FRAME STORE MEMORY CONTROLLER CSR OFFSET

                CPUIOVFLAG=     AVCSR-OØØØØ72
                CPUIDATARDØ=    AVCSR-OØØØØ7Ø
                CPUIDATARD1=    AVCSR-OØØØØ66
                CPUIFSTEST=     AVCSR-OØØØØ64
                CPUISEQSUB=     AVCSR-OØØØØ62
                CPUIMAINT=      AVCSR-OØØØØ6Ø   ; MAINTENANCE REGISTER
                CPUIACR=        AVCSR-OØØØØ56   ; ACCESS CONTROL REGISTER
                CPUIYSTR=       AVCSR-OØØØØ52
                CPUIXSTR=       AVCSR-OØØØØ5Ø
                CPUIYFENCE=     AVCSR-OØØØØ46
                CPUIXFENCE=     AVCSR-OØØØØ44
                CPUIYADDR=      AVCSR-OØØØØ42
                CPUIXADDR=      AVCSR-OØØØØ4Ø
                CPUICOMPØ=      AVCSR-OØØØØ36
                CPUICOMP1=      AVCSR-OØØØØ34
                CPUICOMP2=      AVCSR-OØØØØ32
                CPUICOMP3=      AVCSR-OØØØØ3Ø
                CPUICOMP4=      AVCSR-OØØØØ26
                CPUICOMP5=      AVCSR-OØØØØ24
                CPUICOMP6=      AVCSR-OØØØØ22
                CPUICOMP7=      AVCSR-OØØØØ2Ø
        .SBTTL  Standard tables
;
; Driver prologue table
;
        DPTAB   -                               ; DPT-creation macro
                END=AVEND,-                     ; End of driver label
                ADAPTER=UBA,-                   ; Adapter type
                UCBSIZE=<UCB$KAVUCBLEN>,-       ; Length of UCB
                NAME=AVDRIVER                   ; Driver name
```

AVA FRAME BUFFER I/O DRIVER

```
        DPTSTORE INIT                          ; Start of load
                                               ; initialization table
        DPTSTORE UCB,UCB$BFIPL,B,8             ; Device fork IPL
        DPTSTORE UCB,UCB$BDIPL,B,22            ; Device interrupt IPL=22=BR6
        DPTSTORE UCB,UCB$LDEVCHAR,L,<-  ; Device characteristics
                DEV$MIDV!-                     ;   input device
                DEV$MAVL!-
                DEV$MODV>                      ;   output device
        DPTSTORE UCB,UCB$BDEVCLASS,B,DC$SCOM   ; Device class?
        DPTSTORE UCB,UCB$BDEVTYPE,B,DT$DR11C   ; DEVICE TYPE (NOT REALLY)
        DPTSTORE UCB,UCB$WDEVBUFSIZ,W,- ; Default buffer size
                AVDEFBUFSIZ
        DPTSTORE REINIT                 ; Start of reload
                                               ; initialization table
        DPTSTORE DDB,DDB$LDDT,D,AV$DDT  ; Address of DDT
        DPTSTORE CRB,CRB$LINTD+4,D,-           ; Address of interrupt
                AVINTERRUPT                    ; service routine
        DPTSTORE CRB,-                         ; Address of controller
                CRB$LINTD+VEC$LINITIAL,-       ; initialization routine
                D,AVCONTROLINIT
        DPTSTORE CRB,-                         ; Address of device
                CRB$LINTD+VEC$LUNITINIT,-      ; unit initialization
                D,AVUNITINIT                   ; routine
        DPTSTORE END                           ; End of initialization
                                               ; tables
;
; Driver dispatch table
;
        DDTAB   -                              ; DDT-creation macro
                DEVNAM=AV,-                    ; Name of device
                START=AVSTART,-         ; Start I/O routine
                FUNCTB=AVFUNCTABLE,-           ; FDT address
                CANCEL=AVCANCEL,-              ; Cancel I/O routine
                REGDMP=AVREGDUMP               ; Register dump routine
;
; Function decision table
;
AVFUNCTABLE:                                   ; FDT for driver
        FUNCTAB ,-                             ; Valid I/O functions
                <READVBLK,-                    ; Read virtual
                READLBLK,-                     ; Read logical
                READPBLK,-                     ; Read physical
                WRITEVBLK,-                    ; Write virtual
                WRITELBLK,-                    ; Write logical
                WRITEPBLK>                     ; Write physical
        FUNCTAB ,-                             ; Buffered functions
                <READVBLK,-                    ; Read virtual
                READLBLK,-                     ; Read logical
                READPBLK,-                     ; Read physical
                WRITEVBLK,-                    ; Write virtual
                WRITELBLK,-                    ; Write logical
                WRITEPBLK>                     ; Write physical
        FUNCTAB AVAFDT,-
                <READVBLK,-                    ; Read virtual
                READLBLK,-                     ; Read logical
                READPBLK,-                      ; Read physical
                WRITEVBLK,-                     ; Write virtual
                WRITELBLK,-                     ; Write logical
                WRITEPBLK>                      ; Write physical
        FUNCTAB AVWRITEAVACFDT,-
                <WRITEVBLK,-                    ; Write virtual
                WRITELBLK,-                     ; Write logical
                WRITEPBLK>                      ; Write physical
        FUNCTAB AVREADAVACFDT,-
                <READVBLK,-                     ; Read virtual
                READLBLK,-                      ; Read logical
                READPBLK>                       ; Read physical
                .LONG   -1                      ; SET ALL BITS FOR THE
                .LONG   -1                      ; FDT CATCH ALL ERROR ROUTINE
                .ADDRESS        OOPS
        .SBTTL  AVCONTROLINIT, Controller initialization routine
;++
; AVCONTROLINIT, Readies controller for I/O operations
;
; Functional description:
;
;       The operating system calls this routine in 3 places:
;
;               at system startup
;               during driver loading and reloading
;               during recovery from a power failure
;
; Inputs:
;
;       R4      - address of the CSR (controller status register)
;       R5      - address of the IDB (interrupt data block)
;       R6      - address of the DDB (device data block)
;       R8      - address of the CRB (channel request block)
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R3.
;
;--
AVCONTROLINIT:                  ; Initialize controller
        RSB                             ; Return
        .SBTTL  AVUNITINIT, Unit initialization routine
;++
; AVUNITINIT, Readies unit for I/O operations
;
; Functional description:
;
;       The operating system calls this routine after calling the
;       controller initialization routine:
;
;               at system startup
;               during driver loading
;               during recovery from a power failure
;
; Inputs:
```

```
;
;       R4      - address of the CSR (controller status register)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R3.
;
;--
AVUNITINIT:                             ; Initialize unit
        BISW     UCB$MONLINE, -
                UCB$WSTS(R5)            ; Set unit online
        CLRW    AVCSR(R4)               ; INITIALIZE AVA FRAME BUFFER
        MOVW     1,AVCSR(R4)            ; SET *INIT BIT IN CSR
        RSB                             ; Return
        .SBTTL  AVFDTROUTINE, ON LINE DIGITIZER AVA FDT routine
;++
; AVFDTROUTINE, ON LINE DIGITIZER AVA FDT routine
;
; Functional description:
;
;       SET UP FOR BUFFERED IO ON THE AVA INTERFACE
;
; Inputs:
;
;       RØ-R2   - scratch registers
;       R3      - address of the IRP (I/O request packet)
;       R4      - address of the PCB (process control block)
;       R5      - address of the UCB (unit control block)
;       R6      - address of the CCB (channel control block)
;       R7      - bit number of the I/O function code
;       R8      - address of the FDT table entry for this routine
;       R9-R11  - scratch registers
;       AP      - address of the 1st function dependent QIO parameter
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R2, and
;       R9-R11.
;
;--
;
;       CATCH ALL FDT ERROR ROUTINE
;
OOPS:
        MOVL     SS$ILLIOFUNC,RØ        ; ILLEGAL I/O FUNCTION SPECIFIED
        JSB     GEXE$ABORTIO            ; SO LET'S ABORT
AVAFDT:
        MOVW    P3(AP),UCB$WAVXADDR(R5) ; STARTING X ADDRESS
        MOVW    P4(AP),UCB$WAVYADDR(R5) ; STARTING Y ADDRESS
        MOVW    P5(AP),UCB$WAVCSR(R5)     ; INITIALIZE ACCES CONTROL REGISTER BITS
        MOVW    P6(AP),UCB$WAVACR(R5)     ; SET UP ACR
        RSB
AVWRITEAVACFDT:                 ; WRITE FDT routine
        MOVQ    P1(AP),RØ               ; MOVE BUFFER ADDRESS IN RØ
```

```
                                        ; AND  BUFFER SIZE    IN R1
        TSTL    R1                      ; IF BUFFER SIZE <=Ø WE HAVE PROBLEMS
        BGTR    1S                      ; OTHERWIZE LETS GET ON WITH IT
        MOVL     SS$IVBUFLEN,RØ ; MOVE THE ERROR STATUS INTO RØ
        JMP     GEXE$FINISHIO           ; BAD BUFFER SIZE
1S:     JSB     GEXE$WRITECHK           ; ABORTS AND DOESN'T COME BACK IF IT
                                        ; CAN'T WRITE TO BUFFER
;       MOVL     SS$NOACNT,RØ           ; ********ONLY FOR ERROR CHECKING****
;       JMP     GEXE$FINISHIO           ; ********ONLY FOR ERROR CHECKING****
        PUSHR    M<R2,R3>               ; SAVE R2 AND R3 FROM BUFFRQUOTA
        JSB     GEXE$BUFFRQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
        POPR     M<R2,R3>               ; RESTORE R2 AND R3
        BLBS    RØ,1ØS                  ; IF ERROR WE EXCEEDED QUOTA
11S:    JMP     GEXE$ABORTIO            ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
1ØS:    PUSHR    M<R3>                  ; SAVE IRP ADDRESS FROM ALLOCBUF
        MOVL    R1,R9                   ; SAVE BUFFER SIZE IN R9 JUST FOR GRINS
        ADDL2    12,R1                  ; SAVE BUFFER SIZE TO CHARGE PROCESS
        JSB     GEXE$ALLOCBUF           ; ALLOCATE SOME NON-PAGED POOL FOR THIS
        POPR     M<R3>                  ; RESTORE IRP ADDRESS TO R3
        BLBC    RØ,11S                  ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
        ADDL3   R2, 12,(R2)             ; INIT FIRST LONGWORD OF BUFFER WITH
                                        ; ADDRESS OF DATA AREA
        MOVL    R2,IRP$LSVAPTE(R3)      ; PUT ADDRESS OF SYSTEM BUFFER
        MOVL    P1(AP),4(R2)            ; INIT SECOND LONGWORD WITH USER BUFFER
                                        ;  ADDRESS
        MOVL    PCB$LJIB(R4),RØ ; JET JIB ADDRESS
        SUBL    R9,JIB$LBYTCNT(RØ)      ; CHARGE PROCESS FOR BUFFER SPACE USED
        PUSHR    M<R1,R2,R3,R4,R5>      ; SAVE ALL THESE FOR THE MOVC
        MOVC3   P2(AP),@4(R2),@(R2)     ; MOVE USER BUFFER INTO SYSTEM BUFFER
        POPR     M<R1,R2,R3,R4,R5>      ; RESTORE THESE NOW AFTER MOVC
        MOVW    R9,IRP$WBOFF(R3)        ; NUMBER OF BYTES CHARGED AGAINST
                                        ;  USER'S PROCESS QUOTA
        JMP     GEXE$QIODRVPKT          ; NOW GO QUEUE I/O REQUEST PACKET
AVREADAVACFDT:                  ; READ FDT routine
        SUBW2    IO$READLBLK-IO$READPBLK,-      ; SET I/O FUNCTION CODE IN IRP
                IRP$WFUNC(R3)
        MOVQ    P1(AP),RØ               ; MOVE BUFFER ADDRESS IN RØ
                                        ; AND  BUFFER SIZE    IN R1
        TSTL    R1                      ; IF BUFFER SIZE <=Ø WE HAVE PROBLEMS
        BGTR    51S                     ; OTHERWIZE LETS GET ON WITH IT
        JMP     GEXE$FINISHIO           ; BAD BUFFER SIZE
51S:    JSB     GEXE$READCH             ; ABORTS AND DOESN'T COME BACK IF IT
                                        ; CAN'T WRITE TO BUFFER
        PUSHR    M<RØ,R3>               ; SAVE RØ AND R3 FROM BUFFRQUOTA
        ADDL2    12,R1
        JSB     GEXE$BUFFRQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
        BLBS    RØ,51ØS                 ; IF ERROR WE EXCEEDED QUOTA
511S:   JMP     GEXE$ABORTIO            ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
51ØS:   JSB     GEXE$ALLOCBUF           ; ALLOCATE SOME NON-PAGED POOL FOR THIS
        BLBC    RØ,511S                 ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
        POPR     M<RØ,R3>               ; RESTORE RØ AND R3
        MOVL    R2,IRP$LSVAPTE(R3)      ; PUT ADDRESS OF SYSTEM BUFFER
        MOVW    R1,IRP$WBOFF(R3)        ; BYTE QUOTA CHARGED
        PUSHL   RØ
        MOVL    PCB$LJIB(R4),RØ ; JET JIB ADDRESS
        SUBL    R1,JIB$LBYTCNT(RØ)      ; CHARGE PROCESS FOR BUFFER SPACE USED
        POPL    RØ
        MOVAB   12(R2),(R2)+            ; SAVE DATA AREA ADDRESS
        MOVL    RØ,(R2)                 ; SAVE USER BUFFER ADDRESS
        JMP     G^EXE$QIODRVPKT         ; NOW GO QUEUE I/O REQUEST PACKET
        .SBTTL  AVSTART, Start I/O routine
;++
; AVSTART - Start a transmit, receive data from or to AVA INTERFACE
;
; Functional description:
;
;       START A READ OR WRITE TO ON LINE DIGITIZER AVA INTERFACE
;
; Inputs:
;
;       R3      - address of the IRP (I/O request packet)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       RØ      - 1st longworAd of I/O status: contains status code and
;                 number of bytes transferred
;       R1      - 2nd longword of I/O status: device-dependent
;
;       The routine must preserve all registers except RØ-R2 and R4.
;
;--
AVSTART:                                ; Process an I/O packet
        DSBINT  UCB$B_DIPL(R5)          ; DISABLE INTERRUPTS
        ADDL2    BUFOVRHD,UCB$L_SVAPTE(R5); SKIP SYS BUF HEADER
        REQPCHAN                        ; PUTS CSR ADDRESS IN R4
        MOVZWL  UCB$W_BCNT(R5),R1
        ASHL     -1,R1,R1
        MOVW    R1,UCB$W_AVBYTCNT(R5)
;       MOVW    UCB$W_BCNT(R5),UCB$W_AVBYTCNT(R5); MOVE BYTE COUNT TO
                                                 ;   NEW UCB FIELD
;       CLRW    UCB$W_BCNT(R5)          ; CLEAR UCB BYTE COUNT
;       MOVW    P5(AP),UCB$W_AVCSR(R5)      ; INITIALIZE ACCES CONTROL REGISTER BITS
        MOVZWL  UCB$W_AVCSR(R5),R1
;       BLBS    R1,NOINIT
        BLBS    R1,CHECKSSW
        MOVW    UCB$W_AVCSR(R5),AVCSR(R4)
;       MOVW     1,AVCSR(R4)
        BISW3   UCB$W_AVCSR(R5), 1,AVCSR(R4)
        MOVW     Ø,CPUIMAINT(R4)
        MOVW     O1776,CPUIXFENCE(R4)
        MOVW     O777,CPUIYFENCE(R4)
;       MOVW     OØØØØ35,CPUIACR(R4)
;       MOVW    P6(AP),UCB$W_AVACR(R5)      ; SET UP ACR
        MOVW    UCB$W_AVACR(R5),CPUIACR(R4)
        MOVW    UCB$W_AVXADDR(R5),CPUIXADDR(R4)
        MOVW    UCB$W_AVYADDR(R5),CPUIYADDR(R4)
CHECKSSW:
        BITW     O4ØØØ,UCB$W_AVCSR(R5)    ;TEST FOR MEMORY WINDOW ENABLE
        BEQL    NOINIT
        MOVW    UCB$WAVCSR(R5),AVCSR(R4)
;
;       WE HAVE A REQUEST TO CHECK THE AVA SPECIAL STATUS WORD
;
SSW:
        MOVL    @IRP$LSVAPTE(R3),UCB$LSVAPTE(R5)        ; GET BUFFER ADDRESS
        SETIPL   IPL$POWER              ; CHECK FOR POWER FAIL
;       BBCC     UCB$VPOWER,-
;               UCB$WSTS(R5),-
;               WAITSPREAD
WAITSPREAD:
        MOVW    CPUICOMP3(R4),@UCB$LSVAPTE(R5); READ INPUT DATA REGISTER
        MOVW     1,AVCSR(R4)
        ENBINT
;
;       THIS MOVW READS FROM THE OVERLAY COMPONENT WHICH WE REALLY DON'T
;       HAVE AND SINCE BIT 11 IS SET IN THE AVA CSR MBA16 IS SET WHICH
;       PUTS THE SPECIAL AVA STATUS ON THE MASTER BUS DATA BUS
        BRW     FINISH
NOINIT:
        MOVW    P3(AP),UCB$WAVXADDR(R5) ; STARTING X ADDRESS
;       MOVW    P4(AP),UCB$WAVYADDR(R5) ; STARTING Y ADDRESS
;
        MOVW    UCB$WAVXADDR(R5),CPUIXADDR(R4)
        MOVW    UCB$WAVYADDR(R5),CPUIYADDR(R4)
;       ENBINT
        CMPZV    IRP$VFCODE, IRP$SFCODE,-
                IRP$WFUNC(R3), IO$READPBLK
        BEQL    READD                   ; WANT TO GO READ AVA

;       DSBINT  UCB$BDIPL(R5)           ; DISABLE INTERRUPTS
WRITE:
;       MOVW    @UCB$LSVAPTE(R5),AVCSR(R4)      ; CSR BIT TESTING.............
        MOVW    @UCB$LSVAPTE(R5),CPUICOMPØ(R4)
;       MOVW    AVCSR(R4),UCB$WAVCSR(R5) ; PUT THE CSR IN IOSB STATUS WORD
;       SETIPL   IPL$POWER              ; CHECK FOR POWER FAIL
;       BBCC     UCB$VPOWER,-
;               UCB$WSTS(R5),-
;               WAITWRITE
;       ENBINT
;       RELCHAN
;       MOVZWL   SS$POWERFAIL,RØ
;       REQCOM
WAITWRITE:
;       WFIKPCH AVTIMEOUT, AVTIMEOUTSEC
;;      INCL    UCB$WBCNT(R5)           ; INCREMENT NUMBER OF WORDS TRANSFERED
;       MOVW    AVOUTBUF(R4),UCB$WAVOUTBUF(R5); DEVICE OUTPUT REGISTER
;       MOVW    AVA74(R4),UCB$WAVCSR(R5) ; PUT THE CSR IN IOSB STATUS WORD
;       IOFORK
;       INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
;       INCL    UCB$LSVAPTE(R5)
        ADDL2    2,UCB$LSVAPTE(R5)
;;      DECW    UCB$WAVBYTCNT(R5)       ; DECREMENT BYTE COUNT TO SEE IF DONE
        DECW    UCB$WAVBYTCNT(R5)       ; DECREMENT BYTE COUNT TO SEE IF DONE
;       SUBL2    2,UCB$WAVBYTCNT(R5)
```

```
        BGTR    WRITE
        ENBINT
FINISH: RELCHAN

;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
        INSV    UCB$WBCNT(R5), 16,-      ; Load number of bytes trans-
                 16,RØ                   ; ferred into high word of RØ.
        MOVW     SS$NORMAL,RØ            ; Load a success code into RØ.
;       INSV    UCB$WAVOUTBUF(R5), 16,- ;LOAD OUTBUF IN IOSB(4)
;                16,R1
;       INSV     Ø, 16, 16,R1            ; CLEAR UPPER WORD, IOSB(4)
;       MOVW    UCB$WAVCSR(R5),R1 ; PUT THE INBUF IN IOSB STATUS WORD
        MOVW    UCB$WAVOUTBUF(R5),R1
;       MOVW     Ø,R1
;
; Call I/O postprocessing.
;
COMPLETEIO:                              ; Driver processing is finished.
        REQCOM                           ; Complete I/O.


;
;       READ LOOP
;
READD:  MOVL    @IRP$LSVAPTE(R3),UCB$LSVAPTE(R5)        ; GET BUFFER ADDRESS
;       DSBINT  UCB$BDIPL(R5)            ; DISABLE INTERRUPTS
        SETIPL   IPL$POWER               ; CHECK FOR POWER FAIL
        BBCC     UCB$VPOWER,-
                UCB$WSTS(R5),-
                READ
READ:
;       WFIKPCH AVTIMEOUT, AVTIMEOUTSEC
;       MOVW    AVINBUF(R4),@UCB$LSVAPTE(R5); READ INPUT DATA REGISTER
        MOVW    CPUICOMPØ(R4),@UCB$LSVAPTE(R5); READ INPUT DATA REGISTER

;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
;;      INCL    UCB$WBCNT(R5)            ; INCREMENT NUMBER OF WORDS TRANSFERED
;       MOVW    AVINBUF(R4),UCB$WAVCSR(R5) ; PUT THE INBUF IN IOSB STATUS WORD
;       MOVW    AVOUTBUF(R4),UCB$WAVOUTBUF(R5); DEVICE OUTPUT REGISTER
;       IOFORK
;       INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
;       INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
        ADDL2    2,UCB$LSVAPTE(R5)
;;      DECW    UCB$WAVBYTCNT(R5)        ; DECREMENT BYTE COUNT TO SEE IF DONE
        DECW    UCB$WAVBYTCNT(R5)        ; DECREMENT BYTE COUNT TO SEE IF DONE
;       SUBL2    2,UCB$WAVBYTCNT(R5)
        BGTR    READ
        ENBINT
        BRW     FINISH
;
; Device timeout handling. Return an error status code.
```

```
;
AVTIMEOUT:                                  ; Timeout handling
;       BICW2   <AVCSRMCSRØ>,AVCSR(R4); SET CONTROL LINE Ø LOW
;       MOVW     Ø,AVCSR(R4)
        SETIPL  UCB$BFIPL(R5)               ; Lower to driver fork IPL
        MOVZWL   SS$TIMEOUT,RØ              ; Return error status.
;       MOVL     63,R1                      ; .. STATUS TESTING..
;       MOVW    AVCSR(R4),UCB$WAVCSR(R5)  ; PUT THE CSR IN IOSB STATUS WORD
        MOVW    UCB$WAVCSR(R5),R1 ; PUT THE CSR IN IOSB STATUS WORD
;       INSV    UCB$WAVOUTBUF(R5), 16,-  ;LOAD OUTBUF IN IOSB(4)
;                16,R1
        BRW     COMPLETEIO                  ; Call I/O postprocessing.
        .SBTTL  AVINTERRUPT, Interrupt service routine
;++
; AVINTERRUPT, Analyzes interrupts, processes solicited interrupts
;
; Functional description:
;
;       The sample code assumes either
;
;               that the driver is for a single-unit controller, and
;               that the unit initialization code has stored the
;               address of the UCB in the IDB; or
;
;               that the driver's start I/O routine acquired the
;               controller's channel with a REQPCHANL macro call, and
;               then invoked the WFIKPCH macro to keep the channel
;               while waiting for an interrupt.
;
; Inputs:
;
;        Ø(SP)  - pointer to the address of the IDB (interrupt data
;                 block)
;        4(SP)  - saved RØ
;        8(SP)  - saved R1
;       12(SP)  - saved R2
;       16(SP)  - saved R3
;       2Ø(SP)  - saved R4
;       24(SP)  - saved R5
;       28(SP)  - saved PSL (program status longword)
;       32(SP)  - saved PC
;
;       The IDB contains the CSR address and the UCB address.
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R5.
;
;--
AVINTERRUPT:                                ; Service device interrupt
        MOVL    @(SP)+,R4                   ; Get address of IDB and remove
                                            ; pointer from stack.
        MOVL    IDB$LOWNER(R4),R5           ; Get address of device owner's
                                            ; UCB.
        MOVL    IDB$LCSR(R4),R4 ; Get address of device's CSR.
```

```
;       BICW2    <AVCSRMCSRØ>,AVCSR(R4); SET CONTROL LINE Ø LOW
        BBCC     UCB$VINT,-             ; If device does not expect
                UCB$WSTS(R5),-          ; interrupt, dismiss it.
                UNSOLINTERRUPT
;
; This is a solicited interrupt. Save
; the contents of the device registers in the UCB. NOT NEEDED IN THIS DRIVER
;
;
; Restore control to the main driver.
;
RESTOREDRIVER:                          ; Jump to main driver code.
        MOVL    UCB$LFR3(R5),R3 ; Restore driver's R3 (use a
                                        ; MOVQ to restore R3-R4).
        JSB     @UCB$LFPC(R5)           ; Call driver at interrupt
                                        ; wait address.
;
; Dismiss the interrupt.
;
UNSOLINTERRUPT:                 ; Dismiss unsolicited interrupt.
        POPR     M<RØ,R1,R2,R3,R4,R5>   ; Restore RØ-R5
        REI                             ; Return from interrupt.
        .SBTTL  AVCANCEL, Cancel I/O routine
;++
; AVCANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;       This routine calls IOC$CANCELIO to set the cancel bit in the
;       UCB status word if:
;
;               the device is busy,
;               the IRP's process ID matches the cancel process ID,
;               the IRP channel matches the cancel channel.
;
;       If IOC$CANCELIO sets the cancel bit, then this driver routine
;       does device-dependent cancel I/O fixups.
;
; Inputs:
;
;       R2      - channel index number
;       R3      - address of the current IRP (I/O request packet)
;       R4      - address of the PCB (process control block) for the
;                 process canceling I/O
;       R5      - address of the UCB (unit control block)
;       R8      - cancel reason code, one of:
;                       CAN$CCANCEL     if called through $CANCEL or
;                                       $DALLOC system service
;                       CAN$CDASSGN     if called through $DASSGN system
;                                       service
;                 These reason codes are defined by the $CANDEF macro.
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R3.
```

```
;
;       The routine may set the UCB$MCANCEL bit in UCB$WSTS.
;
;--
AVCANCEL:                                ; Cancel an I/O operation
        JSB     GIOC$CANCELIO            ; Set cancel bit if appropriate.
        BBC      UCB$VCANCEL,-           ; If the cancel bit is not set,
                UCB$WSTS(R5),1ØS         ; just return.
;
; Device-dependent cancel operations go next.
;
;
; Finally, the return.
;
1ØS:
        RSB                              ; Return
        .SBTTL  AVREGDUMP, Device register dump routine
;++
; AVREGDUMP, Dumps the contents of device registers to a buffer
;
; Functional description:
;
;       Writes the number of device registers, and their current
;       contents into a diagnostic or error buffer.
;
; Inputs:
;
;       RØ      - address of the output buffer
;       R4      - address of the CSR (controller status register)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       The routine must preserve all registers except R1-R3.
;
;       The output buffer contains the current contents of the device
;       registers. RØ contains the address of the next empty longword in
;       the output buffer.
;
;--
AVREGDUMP:                               ; Dump device registers
        MOVZBL   AVNUMREGS,(RØ)+         ; Store device register count.
        MOVZWL  UCB$WAVBYTCNT(R5),-      ; Store BYTE count register.
                (RØ)+
        RSB                              ; Return
        .SBTTL  AVEND, End of driver
;++
; Label that marks the end of the driver
;--
AVEND:                                   ; Last location in driver
        .END
```

APPENDIX AJ

ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```
        .TITLE  ODDRIVER - VAX/VMS ON LINE DIGITIZER TAPE CONTROLLER DRIVER (DR11-C)
        .IDENT  'VØ3-ØØ1'
;++
;
; FACILITY:
;
;       VAX/VMS On Line Digitizer Tape controller driver (DR11-C)
;
; ABSTRACT:
;
;       This module contains the driver:
;
;               Tables
;               Controller and unit initialization routines
;               The FDT routine
;               The start I/O routine
;               The interrupt service routine
;               The cancel I/O routine
;               The device register dump routine
;
; AUTHOR:
;
;       S. Richard F. Sims  Aug. 23, 1982
;
; REVISION HISTORY:
;
;--
        .SBTTL  External and local symbol definitions
;
; External symbols
;
        $CANDEF                         ; Cancel reason codes
        $CRBDEF                         ; Channel request block
        $DCDEF                          ; Device classes and types
        $DDBDEF                         ; Device data block
        $DEVDEF                         ; Device characteristics
        $IDBDEF                         ; Interrupt data block
        $IODEF                          ; I/O function codes
        $IPLDEF                         ; Hardware IPL definitions
        $IRPDEF                         ; I/O request packet
```

ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```
        $SSDEF                          ; System status codes
        $UCBDEF                         ; Unit control block
        $VECDEF                         ; Interrupt vector block
        $JIBDEF                         ; JOB INFO BLOCK OFFSET DEFS
        $PCBDEF                         ; PROCESS CONTROL BLOCK OFFSET DEFS
;
; Local symbols
;
;
; Argument list (AP) offsets for device-dependent QIO parameters
;
P1      = Ø                             ; First QIO parameter
P2      = 4                             ; Second QIO parameter
P3      = 8                             ; Third QIO parameter
P4      = 12                            ; Fourth QIO parameter
P5      = 16                            ; Fifth QIO parameter
P6      = 2Ø                            ; Sixth QIO parameter
;
; Other constants
;
ODDEFBUFSIZ     = 1                     ; Default buffer size
ODTIMEOUTSEC    = 1Ø                    ; 1Ø second device timeout
ODNUMREGS       = 3                     ; Device has 3 registers
BUFOVRHD        = 12                    ; SYSTEM BUFFER OVERHEAD FOR BUFFERED I/O
;
; Definitions that follow the standard UCB fields
;
        $DEFINI UCB                     ; Start of UCB definitions
        .=UCB$KLENGTH                   ; Position at end of UCB
$DEF    UCB$WODCSR                      ; Device's CSR register
                        .BLKW   1
$DEF    UCB$WODBYTCNT                   ; Device's BYTE count register
                        .BLKW   1
$DEF    UCB$WODOUTBUF                   ; DEVICE OUTBUF REGISTER
                        .BLKW   1
$DEF    UCB$KODUCBLEN                   ; Length of extended UCB
                        .BLKW   1
;
; Bit positions for device-dependent status field in UCB
;
        $VIELD  UCBCSR,Ø,<-             ; Device status
                <BITZERO,,M>,-          ; First bit
                <BITONE,,M>,-           ; Second bit
                >
        $DEFEND UCB                     ; End of UCB definitions
;
; Device register offsets from CSR address
;
        $DEFINI OD                      ; Start of status definitions
$DEF    ODCSR                           ; Control/status
                        .BLKW   1
;
; Bit positions for device control/status register
;
        VIELD   ODCSR,Ø,<-              ; Control/status register
```

```
                <CSRØ,,M>,-             ; COMMAND BIT Ø
                <CSR1,,M>,-             ; COMMAND BIT 1
                <,3>,-                  ; THREE UNUSED BITS
                <IEB,,M>,-              ; ENABLE REQUEST B INTERRUPTS
                <IEA,,M>,-              ; Enable REQUEST A interrupts
                <REQA,,M>,-             ; UNDER CONTROL OF USER DEVICE
;                                       ;  NORMALLY USED FOR READY INDICATIONS
                <,7>,-                  ; SEVEN Disregarded bits
                <REQB,,M>-              ; UNDER CONTROL OF USER DEVICE
;                                       ;  NORMALLY USED FOR ERROR CONDITIONS
        >
$DEF    ODOUTBUF                        ; OUTPUT BUFFER WORD
                        .BLKW   1
$DEF    ODINBUF                 ; INPUT BUFFER WORD
                        .BLKW   1
        $DEFEND OD                      ; End of device register
                                        ; definitions.
        .SBTTL  Standard tables
;
; Driver prologue table
;
        DPTAB   -                               ; DPT-creation macro
                END=ODEND,-                     ; End of driver label
                ADAPTER=UBA,-                   ; Adapter type
                UCBSIZE=<UCB$K_ODUCBLEN>,-      ; Length of UCB
                NAME=ODDRIVER                   ; Driver name
        DPTSTORE INIT                           ; Start of load
                                                ; initialization table
        DPTSTORE UCB,UCB$B_FIPL,B,8             ; Device fork IPL
        DPTSTORE UCB,UCB$B_DIPL,B,21            ; Device interrupt IPL=21=BR5
        DPTSTORE UCB,UCB$L_DEVCHAR,L,<-  ; Device characteristics
                DEV$M_IDV!-                     ;   input device
                DEV$M_AVL!-
                DEV$M_ODV>                      ;   output device
        DPTSTORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM  ; Device class?
        DPTSTORE UCB,UCB$B_DEVTYPE,B,DT$_DR11C  ; DEVICE TYPE
        DPTSTORE UCB,UCB$W_DEVBUFSIZ,W,- ; Default buffer size
                OD_DEF_BUFSIZ
        DPTSTORE REINIT                  ; Start of reload
                                                ; initialization table
        DPTSTORE DDB,DDB$L_DDT,D,OD$DDT  ; Address of DDT
        DPTSTORE CRB,CRB$L_INTD+4,D,-           ; Address of interrupt
                OD_INTERRUPT                    ; service routine REQ A
        DPTSTORE CRB,CRB$L_INTD2+4,D,-          ; REQ B INTERRUPT ROUTINE
                OD_INTERRUPT                    ;
        DPTSTORE CRB,-                          ; Address of controller
                CRB$L_INTD+VEC$L_INITIAL,-      ; initialization routine
                D,OD_CONTROL_INIT
        DPTSTORE CRB,-                          ; Address of device
                CRB$L_INTD+VEC$L_UNITINIT,-     ; unit initialization
                D,OD_UNIT_INIT                  ; routine
        DPTSTORE END                            ; End of initialization
                                                ; tables
;
; Driver dispatch table
```

ON LINE DIGITIZER TAPE CONTROLLER DRIVER

```
;
        DDTAB   -                               ; DDT-creation macro
                DEVNAM=OD,-                     ; Name of device
                START=ODSTART,-         ; Start I/O routine
                FUNCTB=ODFUNCTABLE,-            ; FDT address
                CANCEL=ODCANCEL,-               ; Cancel I/O routine
                REGDMP=ODREGDUMP                ; Register dump routine
;
; Function decision table
;
ODFUNCTABLE:                                    ; FDT for driver
        FUNCTAB ,-                              ; Valid I/O functions
                <READVBLK,-                     ; Read virtual
                READLBLK,-                      ; Read logical
                READPBLK,-                      ; Read physical
                WRITEVBLK,-                     ; Write virtual
                WRITELBLK,-                     ; Write logical
                WRITEPBLK>                      ; Write physical
        FUNCTAB ,-                              ; Buffered functions
                <READVBLK,-                     ; Read virtual
                READLBLK,-                      ; Read logical
                READPBLK,-                      ; Read physical
                WRITEVBLK,-                     ; Write virtual
                WRITELBLK,-                     ; Write logical
                WRITEPBLK>                      ; Write physical
        FUNCTAB ODWRITEDR11CFDT,-
                <WRITEVBLK,-                    ; Write virtual
                WRITELBLK,-                     ; Write logical
                WRITEPBLK>                      ; Write physical
        FUNCTAB ODREADDR11CFDT,-
                <READVBLK,-                     ; Read virtual
                READLBLK,-                      ; Read logical
                READPBLK>                       ; Read physical
                .LONG   -1                      ; SET ALL BITS FOR THE
                .LONG   -1                      ; FDT CATCH ALL ERROR ROUTINE
                .ADDRESS        OOPS
        .SBTTL  ODCONTROLINIT, Controller initialization routine
;++
; ODCONTROLINIT, Readies controller for I/O operations
;
; Functional description:
;
;       The operating system calls this routine in 3 places:
;
;               at system startup
;               during driver loading and reloading
;               during recovery from a power failure
;
; Inputs:
;
;       R4      - address of the CSR (controller status register)
;       R5      - address of the IDB (interrupt data block)
;       R6      - address of the DDB (device data block)
;       R8      - address of the CRB (channel request block)
;
```

```
; Outputs:
;
;       The routine must preserve all registers except RØ-R3.
;
;--
ODCONTROLINIT:                  ; Initialize controller
        RSB                             ; Return
        .SBTTL  ODUNITINIT, Unit initialization routine
;++
; ODUNITINIT, Readies unit for I/O operations
;
; Functional description:
;
;       The operating system calls this routine after calling the
;       controller initialization routine:
;
;               at system startup
;               during driver loading
;               during recovery from a power failure
;
; Inputs:
;
;       R4      - address of the CSR (controller status register)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R3.
;
;--
ODUNITINIT:                             ; Initialize unit
        BISW     UCB$M_ONLINE, -
                UCB$W_STS(R5)           ; Set unit online
        RSB                             ; Return
        .SBTTL  ODFDTROUTINE, ON LINE DIGITIZER DR11-C FDT routine
;++
; ODFDTROUTINE, ON LINE DIGITIZER DR11-C FDT routine
;
; Functional description:
;
;       SET UP FOR BUFFERED IO ON THIS DR11-C
;
; Inputs:
;
;       RØ-R2   - scratch registers
;       R3      - address of the IRP (I/O request packet)
;       R4      - address of the PCB (process control block)
;       R5      - address of the UCB (unit control block)
;       R6      - address of the CCB (channel control block)
;       R7      - bit number of the I/O function code
;       R8      - address of the FDT table entry for this routine
;       R9-R11  - scratch registers
;       AP      - address of the 1st function dependent QIO parameter
;
; Outputs:
```

```
;
;       The routine must preserve all registers except RØ-R2, and
;       R9-R11.
;
;--
;
;       CATCH ALL FDT ERROR ROUTINE
;
OOPS:
        MOVL     SS$ILLIOFUNC,RØ         ; ILLEGAL I/O FUNCTION SPECIFIED
        JSB     GEXE$ABORTIO             ; SO LET'S ABORT
ODWRITEDR11CFDT:                         ; WRITE FDT routine
        MOVQ    P1(AP),RØ                ; MOVE BUFFER ADDRESS IN RØ
                                         ; AND  BUFFER SIZE    IN R1
        TSTL    R1                       ; IF BUFFER SIZE <=Ø WE HAVE PROBLEMS
        BGTR    1$                       ; OTHERWIZE LETS GET ON WITH IT
        MOVL     SS$IVBUFLEN,RØ ; MOVE THE ERROR STATUS INTO RØ
        JMP     GEXE$FINISHIO            ; BAD BUFFER SIZE
1$:     JSB     GEXE$WRITECHK            ; ABORTS AND DOESN'T COME BACK IF IT
                                         ; CAN'T WRITE TO BUFFER
;       MOVL     SS$NOACNT,RØ            ; ********ONLY FOR ERROR CHECKING****
;       JMP     GEXE$FINISHIO            ; ********ONLY FOR ERROR CHECKING****
        PUSHR    M<R2,R3>                ; SAVE R2 AND R3 FROM BUFFRQUOTA
        JSB     GEXE$BUFFRQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
        POPR     M<R2,R3>                ; RESTORE R2 AND R3
        BLBS    RØ,1Ø$                   ; IF ERROR WE EXCEEDED QUOTA
11$:    JMP     GEXE$ABORTIO             ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
1Ø$:    PUSHR    M<R3>                   ; SAVE IRP ADDRESS FROM ALLOCBUF
        MOVL    R1,R9                    ; SAVE BUFFER SIZE IN R9 JUST FOR GRINS
        ADDL2    12,R1                   ; SAVE BUFFER SIZE TO CHARGE PROCESS
        JSB     GEXE$ALLOCBUF            ; ALLOCATE SOME NON-PAGED POOL FOR THIS
        POPR     M<R3>                   ; RESTORE IRP ADDRESS TO R3
        BLBC    RØ,11$                   ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
        ADDL3   R2, 12,(R2)              ; INIT FIRST LONGWORD OF BUFFER WITH
                                         ; ADDRESS OF DATA AREA
        MOVL    R2,IRP$LSVAPTE(R3)       ; PUT ADDRESS OF SYSTEM BUFFER
        MOVL    P1(AP),4(R2)             ; INIT SECOND LONGWORD WITH USER BUFFER
                                         ;  ADDRESS
        MOVL    PCB$LJIB(R4),RØ ; JET JIB ADDRESS
        SUBL    R9,JIB$LBYTCNT(RØ)       ; CHARGE PROCESS FOR BUFFER SPACE USED
        PUSHR    M<R1,R2,R3,R4,R5>       ; SAVE ALL THESE FOR THE MOVC
        MOVC3   P2(AP),@4(R2),@(R2)      ; MOVE USER BUFFER INTO SYSTEM BUFFER
        POPR     M<R1,R2,R3,R4,R5>       ; RESTORE THESE NOW AFTER MOVC
        MOVW    R9,IRP$WBOFF(R3)         ; NUMBER OF BYTES CHARGED AGAINST
                                         ;  USER'S PROCESS QUOTA
        JMP     GEXE$QIODRVPKT           ; NOW GO QUEUE I/O REQUEST PACKET
ODREADDR11CFDT:                 ; READ FDT routine
        SUBW2    IO$READLBLK-IO$READPBLK,-       ; SET I/O FUNCTION CODE IN IRP
                IRP$WFUNC(R3)
        MOVQ    P1(AP),RØ                ; MOVE BUFFER ADDRESS IN RØ
                                         ; AND  BUFFER SIZE    IN R1
        TSTL    R1                       ; IF BUFFER SIZE <=Ø WE HAVE PROBLEMS
        BGTR    51$                      ; OTHERWIZE LETS GET ON WITH IT
        JMP     GEXE$FINISHIO            ; BAD BUFFER SIZE
51$:    JSB     GEXE$READCHK             ; ABORTS AND DOESN'T COME BACK IF IT
```

```
                                                ; CAN'T WRITE TO BUFFER
        PUSHR    M<RØ,R3>                       ; SAVE RØ AND R3 FROM BUFFRQUOTA
        ADDL2    12,R1
        JSB     GEXE$BUFFRQUOTA ; CHECK TO SEE IF QUOTA CAN HANDLE THIS
        BLBS    RØ,51ØS                         ; IF ERROR WE EXCEEDED QUOTA
511S:   JMP     GEXE$ABORTIO                    ; GO TELL HIM ABOUT THE ERROR AND DON'T COME BACK
51ØS:   JSB     GEXE$ALLOCBUF                   ; ALLOCATE SOME NON-PAGED POOL FOR THIS
        BLBC    RØ,511S                         ; IF ERROR INSUFFICIENT MEMORY AVAILABLE
        POPR     M<RØ,R3>                       ; RESTORE RØ AND R3
        MOVL    R2,IRP$LSVAPTE(R3)              ; PUT ADDRESS OF SYSTEM BUFFER
        MOVW    R1,IRP$WBOFF(R3)                ; BYTE QUOTA CHARGED
        PUSHL   RØ
        MOVL    PCB$LJIB(R4),RØ ; JET JIB ADDRESS
        SUBL    R1,JIB$LBYTCNT(RØ)              ; CHARGE PROCESS FOR BUFFER SPACE USED
        POPL    RØ
        MOVAB   12(R2),(R2)+                    ; SAVE DATA AREA ADDRESS
        MOVL    RØ,(R2)                         ; SAVE USER BUFFER ADDRESS
        JMP     GEXE$QIODRVPKT                  ; NOW GO QUEUE I/O REQUEST PACKET
        .SBTTL  ODSTART, Start I/O routine
;++
; ODSTART - Start a transmit, receive data from or to dr11-c
;
; Functional description:
;
;       START A READ OR WRITE TO ON LINE DIGITIZER DR11-C
;
; Inputs:
;
;       R3      - address of the IRP (I/O request packet)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       RØ      - 1st longword of I/O status: contains status code and
;                 number of bytes transferred
;       R1      - 2nd longword of I/O status: device-dependent
;
;       The routine must preserve all registers except RØ-R2 and R4.
;
;--
ODSTART:                                        ; Process an I/O packet
        ADDL2    BUFOVRHD,UCB$LSVAPTE(R5); SKIP SYS BUF HEADER
        REQPCHAN                                ; PUTS CSR ADDRESS IN R4
        MOVW    UCB$WBCNT(R5),UCB$WODBYTCNT(R5); MOVE BYTE COUNT TO
                                                ;    NEW UCB FIELD
        CLRW    UCB$WBCNT(R5)                   ; CLEAR UCB BYTE COUNT
        CLRW    ODCSR(R4)                       ; CLEAR CSR
;       EXTZV    IRP$VFCODE, IRP$SFCODE,-
;       IRP$WFUNC(R3),R2
;       CMPL     IO$READPBLK,R2
        CMPZV    IRP$VFCODE, IRP$SFCODE,-
        IRP$WFUNC(R3), IO$READPBLK
        BEQL    READD                           ; WANT TO GO READ DR11-C
;       BRB     READD
WRITE:
        DSBINT  UCB$BDIPL(R5)           ; DISABLE INTERRUPTS
        MOVW    @UCB$LSVAPTE(R5),ODOUTBUF(R4); PUT DATA INTO DEVICE OUTPUT REGISTER
        BISW2    <ODCSRMIEA>,-   ;+ODCSRMCSRØ
                ODCSR(R4)               ;ENABLE DEVICE TO INTERRUPT
        SETIPL   IPL$POWER              ; CHECK FOR POWER FAIL
        BBCC     UCB$VPOWER,-
                UCB$WSTS(R5),-
                WAITWRITE
        ENBINT
        RELCHAN
        MOVZWL   SS$POWERFAIL,RØ
        REQCOM
WAITWRITE:
        WFIKPCH ODTIMEOUT, ODTIMEOUTSEC
        INCL    UCB$WBCNT(R5)           ; INCREMENT NUMBER OF WORDS TRANSFERED
;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
        MOVW    ODOUTBUF(R4),UCB$WODOUTBUF(R5); DEVICE OUTPUT REGISTER
        MOVW    ODINBUF(R4),UCB$WODCSR(R5) ; PUT THE INBUF IN IOSB STATUS WORD
        IOFORK
        INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
        INCL    UCB$LSVAPTE(R5)
        DECW    UCB$WODBYTCNT(R5)       ; DECREMENT BYTE COUNT TO SEE IF DONE
        DECW    UCB$WODBYTCNT(R5)       ; DECREMENT BYTE COUNT TO SEE IF DONE
        BGTR    WRITE
FINISH: RELCHAN

        INSV    UCB$WBCNT(R5), 16,-     ; Load number of bytes trans-
                 16,RØ                  ; ferred into high word of RØ.
        MOVW     SS$NORMAL,RØ           ; Load a success code into RØ.
        INSV    UCB$WODOUTBUF(R5), 16,- ;LOAD OUTBUF IN IOSB(4)
                 16,R1
;       INSV    Ø, 16, 16,R1            ; CLEAR UPPER WORD. IOSB(4)
        MOVW    UCB$WODCSR(R5),R1 ; PUT THE INBUF IN IOSB STATUS WORD
;
; Call I/O postprocessing.
;
COMPLETEIO:                             ; Driver processing is finished.
        REQCOM                          ; Complete I/O.
;
;       READ LOOP
;
READD:  MOVL    @IRP$LSVAPTE(R3),UCB$LSVAPTE(R5)        ; GET BUFFER ADDRESS
READ:
        DSBINT  UCB$BDIPL(R5)           ; DISABLE INTERRUPTS
GO:     BISW2    <ODCSRMIEB+ODCSRMCSRØ>,-       ;ODCSRMIEA+
                ODCSR(R4)               ;ENABLE DEVICE TO INTERRUPT
        SETIPL   IPL$POWER              ; CHECK FOR POWER FAIL
        BBCC     UCB$VPOWER,-
                UCB$WSTS(R5),-
                WAITREAD
        ENBINT
        RELCHAN
        MOVZWL  SS$POWERFAIL,RØ
        REQCOM
WAITREAD:
        WFIKPCH ODTIMEOUT, ODTIMEOUTSEC
        MOVW    ODINBUF(R4),@UCB$LSVAPTE(R5); READ INPUT DATA REGISTER
;
; After a transfer completes successfully, return the number of bytes
; transferred and a success status code.
;
        INCL    UCB$WBCNT(R5)           ; INCREMENT NUMBER OF WORDS TRANSFERED
        MOVW    ODINBUF(R4),UCB$WODCSR(R5) ; PUT THE INBUF IN IOSB STATUS WORD
        MOVW    ODOUTBUF(R4),UCB$WODOUTBUF(R5); DEVICE OUTPUT REGISTER
        IOFORK
        INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
        INCL    UCB$LSVAPTE(R5) ; INCREMENT SYSTEM DATA AREA ADDRESS
        DECW    UCB$WODBYTCNT(R5)       ; DECREMENT BYTE COUNT TO SEE IF DONE
        DECW    UCB$WODBYTCNT(R5)       ; DECREMENT BYTE COUNT TO SEE IF DONE
        BGTR    READ
        BRW     FINISH
;
; Device timeout handling. Return an error status code.
;
ODTIMEOUT:                              ; Timeout handling
        MOVW     Ø,@UCB$LSVAPTE(R5)     ; NO DATA PUT OUT
        BICW2    <ODCSRMCSRØ>,ODCSR(R4); SET CONTROL LINE Ø LOW
        SETIPL  UCB$BFIPL(R5)           ; Lower to driver fork IPL
        MOVZWL   SS$TIMEOUT,RØ          ; Return error status.
;       MOVL     63,R1                  ; .. STATUS TESTING..
        MOVW    UCB$WODCSR(R5),R1 ; PUT THE CSR IN IOSB STATUS WORD
        BRB     COMPLETEIO              ; Call I/O postprocessing.
        .SBTTL  ODINTERRUPT, Interrupt service routine
;++
; ODINTERRUPT, Analyzes interrupts, processes solicited interrupts
;
; Functional description:
;
;       The sample code assumes either
;
;               that the driver is for a single-unit controller, and
;               that the unit initialization code has stored the
;               address of the UCB in the IDB; or
;
;               that the driver's start I/O routine acquired the
;               controller's channel with a REQPCHANL macro call, and
;               then invoked the WFIKPCH macro to keep the channel
;               while waiting for an interrupt.
;
; Inputs:
;
;         Ø(SP)  - pointer to the address of the IDB (interrupt data
;                  block)
;         4(SP)  - saved RØ
;         8(SP)  - saved R1
;        12(SP)  - saved R2
;        16(SP)  - saved R3
;       2Ø(SP)  - saved R4
;       24(SP)  - saved R5
;       28(SP)  - saved PSL (program status longword)
;       32(SP)  - saved PC
;
;       The IDB contains the CSR address and the UCB address.
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R5.
;
;--
ODINTERRUPT:                            ; Service device interrupt
        MOVL    @(SP)+,R4               ; Get address of IDB and remove
                                        ; pointer from stack.
        MOVL    IDB$L_OWNER(R4),R5       ; Get address of device owner's
                                        ; UCB.
        MOVL    IDB$L_CSR(R4),R4 ; Get address of device's CSR.
        BICW2    <ODCSRMCSRØ>,ODCSR(R4); SET CONTROL LINE Ø LOW
        BBCC     UCB$V_INT,-             ; If device does not expect
                UCB$W_STS(R5),-          ; interrupt, dismiss it.
                UNSOLINTERRUPT
;
; This is a solicited interrupt. Save
; the contents of the device registers in the UCB. NOT NEEDED IN THIS DRIVER
;
;
; Restore control to the main driver.
;
RESTOREDRIVER:                          ; Jump to main driver code.
        MOVL    UCB$L_FR3(R5),R3 ; Restore driver's R3 (use a
                                        ; MOVQ to restore R3-R4).
        JSB     @UCB$L_FPC(R5)          ; Call driver at interrupt
                                        ; wait address.
;
; Dismiss the interrupt.
;
UNSOLINTERRUPT:                 ; Dismiss unsolicited interrupt.
        POPR     M<RØ,R1,R2,R3,R4,R5>    ; Restore RØ-R5
        REI                             ; Return from interrupt.
        .SBTTL  ODCANCEL, Cancel I/O routine
;++
; ODCANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;       This routine calls IOC$CANCELIO to set the cancel bit in the
;       UCB status word if:
;
;               the device is busy,
;               the IRP's process ID matches the cancel process ID,
;               the IRP channel matches the cancel channel.
;
;       If IOC$CANCELIO sets the cancel bit, then this driver routine
;       does device-dependent cancel I/O fixups.
```

```
;
; Inputs:
;
;       R2      - channel index number
;       R3      - address of the current IRP (I/O request packet)
;       R4      - address of the PCB (process control block) for the
;                 process canceling I/O
;       R5      - address of the UCB (unit control block)
;       R8      - cancel reason code, one of:
;                       CAN$C_CANCEL    if called through $CANCEL or
;                                       $DALLOC system service
;                       CAN$C_DASSGN    if called through $DASSGN system
;                                       service
;                 These reason codes are defined by the $CANDEF macro.
;
; Outputs:
;
;       The routine must preserve all registers except RØ-R3.
;
;       The routine may set the UCB$M_CANCEL bit in UCB$W_STS.
;
;--
ODCANCEL:                               ; Cancel an I/O operation
        JSB     G^IOC$CANCELIO          ; Set cancel bit if appropriate.
        BBC      #UCB$V_CANCEL,-        ; If the cancel bit is not set,
                UCB$W_STS(R5),1Ø$       ; just return.
;
; Device-dependent cancel operations go next.
;
;
; Finally, the return.
;
1Ø$:
        RSB                             ; Return
        .SBTTL  ODREGDUMP, Device register dump routine
;++
; ODREGDUMP, Dumps the contents of device registers to a buffer
;
; Functional description:
;
;       Writes the number of device registers, and their current
;       contents into a diagnostic or error buffer.
;
; Inputs:
;
;       RØ      - address of the output buffer
;       R4      - address of the CSR (controller status register)
;       R5      - address of the UCB (unit control block)
;
; Outputs:
;
;       The routine must preserve all registers except R1-R3.
;
;       The output buffer contains the current contents of the device
;       registers. RØ contains the address of the next empty longword in
;       the output buffer.
;
;--
ODREGDUMP:                                    ; Dump device registers
        MOVZBL   ODNUMREGS,(RØ)+              ; Store device register count.
        MOVZWL  UCBSWODBYTCNT(R5),-           ; Store BYTE count register.
                (RØ)+
        RSB                                   ; Return
        .SBTTL  ODEND, End of driver
;++
; Label that marks the end of the driver
;--
ODEND:                                        ; Last location in driver
        .END
```

DISTRIBUTION

	No. Copies
US Army Materiel System Analysis Activity ATTN: DRXSY-MP Aberdeen Proving Ground, MD 21005	1
DRSMI-R, Dr. McCorkle	1
Dr. Rhoades	1
-RE, Mr. Lindberg	1
-RES	15
-RPT, Record	1
-RPR, Reference	15
-LP, Mr. Voigt	1